IJDAR (2010) 13:121–131
DOI 10.1007/s10032-009-0111-y

ORIGINAL PAPER

# A robust model for on-line handwritten japanese text recognition

**Bilan Zhu · Xiang-Dong Zhou · Cheng-Lin Liu · Masaki Nakagawa**

Received: 7 April 2009 / Revised: 18 November 2009 / Accepted: 14 December 2009 / Published online: 16 January 2010


**Abstract** This paper describes a robust context integration model for on-line handwritten Japanese text recognition. Based on string class probability approximation, the proposed method evaluates the likelihood of candidate segmentation–recognition paths by combining the scores of character recognition, unary and binary geometric features, as well as linguistic context. The path evaluation criterion can flexibly combine the scores of various contexts and is insensitive to the variability in path length, and so, the optimal segmentation path with its string class can be effectively found by Viterbi search. Moreover, the model parameters are estimated by the genetic algorithm so as to optimize the holistic string recognition performance. In experiments on horizontal text lines extracted from the TUAT Kondate database, the proposed method achieves the segmentation rate of 0.9934 that corresponds to a f-measure and the character recognition rate of 92.80%.

**Keywords** On-line Japanese text recognition · String recognition · Integrated segmentation and recognition · Path evaluation

B. Zhu (✉) · X.-D. Zhou · M. Nakagawa
Department of Computer Science, Tokyo University of Agriculture and Technology (TUAT), Tokyo 184-8588, Japan
e-mail: zhubilan@cc.tuat.ac.jp

X.-D. Zhou
e-mail: xdzhou@cc.tuat.ac.jp

M. Nakagawa
e-mail: nakagawa@cc.tuat.ac.jp

C.-L. Liu
National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, 100190 Beijing, China
e-mail: liucl@nlpr.ia.ac.cn

## 1 Introduction

With pen input devices of large writing areas such as tablet PCs, electronic whiteboards and digital pens (e.g., Anoto pen), people tend to write texts continuously with little constraints. This urges the need of handwritten text (character string) recognition. Compared to isolated character recognition, handwritten text recognition faces the difficulty of character segmentation because characters cannot be reliably segmented before they are recognized. Moreover, in continuous handwriting, characters tend to be written more cursively.

Character segmentation of continuous Chinese/Japanese handwriting is difficult due to the facts that the space between characters is not obvious, many characters comprise multiple radicals with internal gaps, and some characters are connected in cursive writing. Dissection methods (such as [1–4]) attempt to segment characters solely according to geometric layout features (gaps, character size/position and inter-relationship). Without character recognition cues and linguistic context, characters cannot be segmented unambiguously by dissection. A feasible way to overcome the ambiguity of segmentation is the so-called integrated segmentation and recognition [5], which is dichotomized into implicit segmentation and explicit segmentation [6]. Implicit segmentation methods (also called segmentation-free methods [7]), mostly combined with hidden Markov model (HMM)-based recognition, simply slice the string pattern into frames of equal length and label the sliced frames (primitive segments), which are concatenated into characters during recognition. Such methods do not incorporate the character shape information sufficiently. Explicit segmentation, attempting to split character patterns at their true boundaries and to label the split character patterns, can better utilize the character shapes into recognition. It is usually accomplished in two steps: over-segmentation and path evaluation-search. The string pattern

is over-segmented into primitive segments such that each segment composes a single character or a part of a character. The segments are combined to generate candidate character patterns (forming a candidate lattice [8]), which are evaluated by character recognition incorporating geometrics and linguistic context.

In over-segmentation-based string recognition, how to evaluate the candidate characters (lying on paths in the candidate lattice) is a key issue. A desirable criterion should make the path of correct segmentation have the largest score. Unlike HMM-based recognition that classifies a unique sequence of feature vectors (each for a frame) on a string, the candidate lattice of over-segmentation has paths of different lengths, each corresponding to a different sequence of feature vectors, thus the comparison of different paths cannot be based on the Bayesian decision theory as for HMM-based recognition. Instead, candidate character recognition and context scores are heuristically combined to evaluate the paths. Such heuristic evaluation criteria can be divided into summation-based ones [9–13] and normalization-based ones [5,14,15]. A summation criterion is the summation of character-wise log-likelihood or the product of probabilistic likelihood. Since the likelihood measure is usually smaller than one, the summation (product) criterion is often biased to paths with fewer characters, and so, tends to over-merge characters. On the other hand, the normalized criterion, obtained by dividing the summation criterion by the number of segmented characters (segmentation length), tends to over-split characters. Another problem of normalized criterion is that the optimal path is not guaranteed by Viterbi search or dynamic programming (DP), because the criterion is not monotonic with the extension of path length.

To better utilize the character shape information in HMM-based recognition while preserving the monotonicity of path evaluation criterion, the variable duration HMM of Chen et al. [16] obtains the state emission probability on a candidate character formed by concatenating multiple frames and replaces the emission probabilities of all these frames. In effect, this corresponds to weighting each candidate character with its number of primitive segments in over-segmentation-based recognition, and leads to improved recognition accuracy and search efficiency [17]. The over-segmentation-based method proposed in Yu et al. [17] uses the number of primitive segments to weight the character recognition scores in a summation criterion to overcome the effect of segmentation length. However, it only weights the character recognition score, and does not explain why the scores of geometric features and linguistic context are not weighted. The path evaluation criteria in Yu et al. [17] cannot be derived from either labeling primitive segments or labeling character patterns. So, we need a method to decide whether to weight each factor using the number of primitive segments or not and give the weighting degree automatically.

In this paper, we present a robust context integration model for on-line handwritten Japanese text recognition. By labeling primitive segments, the proposed method not only can integrate the character shape information into recognition by introducing some adjustable parameters, but also is insensitive to the number of segmented character patterns because the summation is over the primitive segments. Moreover, by optimizing with the genetic algorithm (GA), we can control whether to weight each factor using the number of primitive segments or not and get the weighting degree automatically. The proposed model evaluates the likelihood of candidate segmentation and its string class by combining the scores of character recognition, geometric features (character pattern sizes, inner gaps, single-character positions, pair-character positions, candidate segmentation points), and linguistic context. With the proposed path evaluation criterion, the optimal path can be efficiently found by Viterbi search. Experimental results on horizontal text lines extracted from the TUAT database of *HANDS-Kondate_t_bf-2001-11* (hereafter, Kondate) [18] demonstrate the superiority of our proposed string recognition model.

The rest of this paper is organized as follows: Sect. 2 gives an overview of our handwritten text recognition system. Section 3 describes the over-segmentation scheme and Sect. 4 details the string recognition model. Section 5 describes the parameter optimization method. Section 6 presents the experimental results and Sect. 7 offers our concluding remarks.

## 2 Processing flow

For on-line handwritten Japanese text recognition, our integrated segmentation and recognition system has three major steps. The input is a handwritten character string or text line composed of a sequence of strokes.

**Step 1: over-segmentation.** Each off-stroke (pen lift between two consecutive strokes) is classified into two classes (segmentation point (SP) and non-segmentation point (NSP)) or undecided according to some geometric features. A segmentation point separates two characters at the off-stroke, while a non-segmentation point indicates the off-stroke is within a character. The off-strokes with low classification confidence are "undecided" points. The group of consecutive strokes between two adjacent segmentation/undecided points is a primitive segment, and one or more consecutive primitive segments form a candidate character pattern.

**Step 2: candidate lattice construction.** By character classification, each candidate character pattern is associated with a number of candidate classes with confidence scores. The combination of all candidate patterns and character

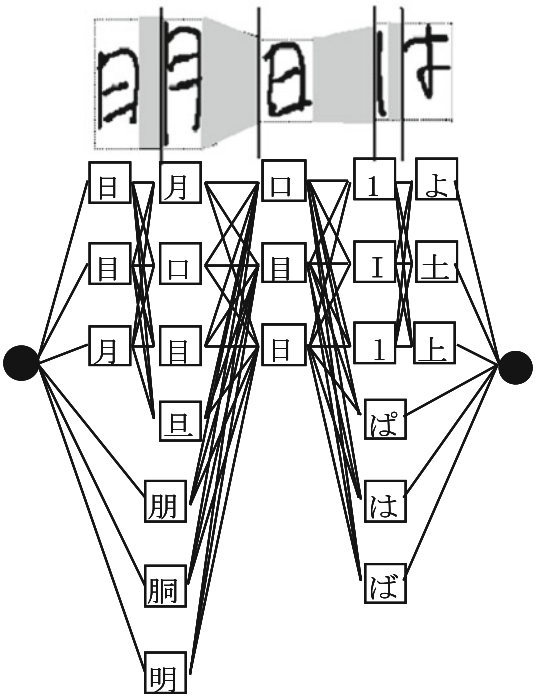


**Fig. 1** Segmentation–recognition candidate lattice

classes is represented by a segmentation–recognition candidate lattice (Fig. 1), where each arc denotes a segmentation point and each node denotes a character class assigned to a candidate pattern.

**Step 3: string recognition.** The segmentation paths and corresponding string classes in the candidate lattice are evaluated by combining the scores of candidate characters and between-character compatibilities (geometric and linguistic contexts). By searching the candidate lattice with the Viterbi algorithm, the optimal path with maximum score gives the final result of character segmentation and recognition. We intend to improve the path evaluation criterion such that the optimal path better corresponds to the correct segmentation and recognition.

## 3 Over-segmentation

We previously proposed an over-segmentation method using an SVM classifier to classify off-strokes into segmentation and non-segmentation points [18]. It extracts multi-dimensional features such as the distance and overlap between adjacent strokes from off-strokes, and the SVM classifier gives fairly high classification accuracy. However, this method often misclassifies true segmentation points as non-segmentation ones when adjacent strokes are overlapping heavily, and misclassifies non-segmentation points as segmentation ones when adjacent strokes are spaced largely.

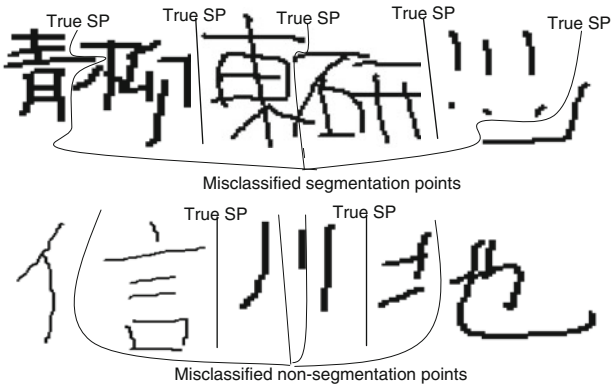



**Fig. 2** Examples of misclassified off-strokes

Some examples of misclassification are shown in Fig. 2. It is hence necessary to leave off-strokes undecided when they cannot be classified reliably. Reducing misclassification via un-decision of some off-strokes can improve the string recognition rate though undecided segmentation points complicate the candidate lattice and consequently the computation of string recognition.

To minimize un-decision and mean while minimize the misclassification between segmentation and non-segmentation points, we herein use an improved two-stage classification scheme. First, we use two geometric features to classify the off-strokes into non-segmentation points and hypothetical ones. Then, an SVM classifier is used to decide some of the hypothetical segmentation points as segmentation points. The process is detailed as follows.

### 3.1 Hypothetical segmentation

We generate hypothetical segmentation points by extracting two features for each off-stroke: horizontal distance and intersecting length.

The horizontal distance feature has been used in Zhu and Nakagawa [18]. It is calculated from two bounding boxes, one for all the strokes preceding the off-stroke (denoted by $BB_{p\_\mathrm{all}}$) and one for the succeeding strokes (denoted by $BB_{s\_\mathrm{all}}$). A distance $DB_x$ is defined as

$$DB_x = left\_bound(BB_{s\_\mathrm{all}}) - right\_bound(BB_{p\_\mathrm{all}}) \tag{1}$$

The horizontal distance feature $f_d$ is calculated by

$$f_d = DB_x/acs \tag{2}$$

where $acs$ is the average character size, which is estimated by measuring the longer side length of the bounding box of each stroke, sorting the lengths of all the strokes and taking the average of the larger 1/3 of them.

For calculating the intersecting length feature, consider the group of all strokes preceding the off-stroke (denoted by

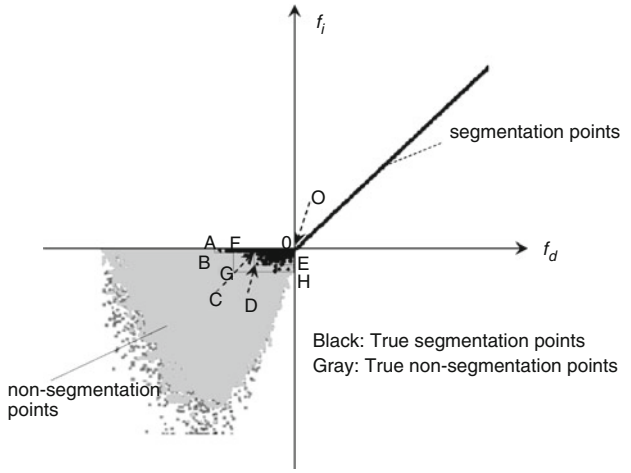



**Fig. 3** Setting thresholds for hypothetical segmentation

$S_{p\_\text{all}}$) and the group of all succeeding strokes (denoted by $S_{s\_\text{all}}$). For a pair of stroke $s_p \in S_{p\_\text{all}}$ and $s_s \in S_{s\_\text{all}}$, if they intersect at a point $p$, calculate the length on $s_p$ from $p$ to the right end of $s_p$ (denoted by $l_p$) and the length on $s_s$ from $p$ to the left end of $s_s$ (denoted by $l_s$). The intersecting length between $s_p$ and $s_s$ is defined as

$$l(s_p, s_s) = \begin{cases} -\min(l_p, l_s)/\max \\ \quad (\text{length and } s_s, \text{length of } s_s), & \text{if } s_p \text{ and } s_s \text{ intersect} \\ 0, \quad \text{otherwise} \end{cases} \tag{3}$$

and accumulated as

$$L_{\text{sum}} = \sum_{s_p \in S_{p\_\text{all}}} \sum_{s_s \in S_{s\_\text{all}}} l(s_p, s_s) \tag{4}$$

The overall intersecting length $f_i$ for an off-stroke is then defined as

$$f_i = \begin{cases} L_{\text{sum}}, & \text{if } L_{\text{sum}} < 0 \\ f_d, & \text{else if } f_d > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Figure 3 shows the distribution of the horizontal distance feature and the intersecting length feature on off-strokes of a set of training string samples. It is shown that segmentation points and non-segmentation ones are well separated by these two features. As shown in Fig. 3, we classify off-strokes as hypothetical segmentation (undecided) points if their values of horizontal distance feature are greater than 0 or they are in the area of OABCDE, and as non-segmentation points otherwise. After this, we modify the classified non-segmentation points between two successive hypothetical segmentation points in the area of OFGH as undecided points, if the width between the two successive hypothetical segmentation points divided by *acs* is greater than a threshold.

Up to now, the off-strokes are classified into non-segmentation points and hypothetical segmentation points. The non-segmentation points are excluded from further consideration (the string pattern cannot be split at a non-segmentation point), while the hypothetical segmentation points are further classified using an SVM classifier.

### 3.2 SVM classification

The hypothetical segmentation points are classified using an SVM classifier on 20 geometric features, among which 18 have been presented in Zhu and Nakagawa [18] as shown in Appendix. Another two features are the intersecting length feature $f_i$ defined in Sect. 3.1 and a newly introduced width feature. The width feature of a hypothetical segmentation point is defined as the width of a box bounding the strokes from the immediately preceding hypothetical segmentation point to the immediately succeeding one, and divided by the average character size *acs*.

We train the SVM using training patterns of off-strokes with the target value of segmentation points set to 1 and that of non-segmentation points to $-1$. On hypothetical segmentation points of test string patterns, the SVM outputs are transformed to probability values (as detailed in Sect. 4.2), which are then combined into the criterion of candidate segmentation-recognition paths.

From SVM classification, we can select some hypothetical segmentation points as segmentation points for which the SVM output is greater than a threshold and the width feature values are greater than a threshold. At the decided segmentation points, the adjacent primitive segments cannot be merged to form candidate character patterns. The reduction in hypothetical segmentation points simplifies the candidate segmentation lattice and improves the efficiency of string recognition.

Keeping all the remaining off-strokes (hypothetical segmentation points) as undecided, segmentation points will incur computation burden. The first stage employs a simple heuristics to eliminate false segmentation boundaries. The SVM classifier increases processing time, but it eliminates them further and improve the recognition rate when it is reflected into the path evaluation function [13].

## 4 String recognition model

In the candidate lattice, it is inappropriate to score the paths using posterior probabilities of characters, because different paths may have different numbers of characters. We herein present an evaluation model that combines multiple features and is theoretically independent of the length of segmentation paths.

### 4.1 Path evaluation

Representing a character string pattern as a sequence of primitive segments: $\mathbf{X} = s_1, \ldots, s_m$, which is partitioned into character patterns $\mathbf{Z} = z_1, \ldots, z_n$, where each candidate pattern contains $k_i$ primitive segments: $z_i = s_{j_i}, \ldots, s_{j_i+k_i-1}$. The segmented character patterns are assigned classes $\mathbf{C} = C_1, \ldots, C_n$. To evaluate the score of string $\mathbf{X}$ in respect to string class $\mathbf{C}$, we extract features for scoring the primitive segments (or candidate patterns) and between-segment (or between-character) compatibilities. The features are listed below:

- Bounding box feature $b_i$
- Inner gap feature $q_i$
- Shape feature $s_i$ or $z_i$
- Unary position feature $p_i^u$ of single segment (or character)
- Binary position feature $p_i^b$ between adjacent segments (or characters)
- Between-segment gap feature $g_i$, which is to be classified as segmentation point or non-segmentation point.

Denoting by $\mathbf{b}$, $\mathbf{q}$, $\mathbf{X}$, $\mathbf{p}^u$, $\mathbf{p}^b$, $\mathbf{g}$ for the sequences of features of primitive segments, the posterior probability of string class is given by:

$$P(\mathbf{C}|\mathbf{X}) = P\left(\mathbf{C}|\mathbf{b}, \mathbf{q}, \mathbf{X}, \mathbf{p^u}, \mathbf{p^b}, \mathbf{g}\right) = \frac{p\left(\mathbf{b}, \mathbf{q}, \mathbf{X}, \mathbf{p^u}, \mathbf{p^b}, \mathbf{g}|\mathbf{C}\right) P(\mathbf{C})}{p\left(\mathbf{b}, \mathbf{q}, \mathbf{X}, \mathbf{p^u}, \mathbf{p^b}, \mathbf{g}\right)} \tag{6}$$

Omitting the string class-independent denominator and reasonably assuming independence between different features, the string class can be equivalently evaluated by

$$f(\mathbf{X}, \mathbf{C}) = \log p\left(\mathbf{b}, \mathbf{q}, \mathbf{X}, \mathbf{p^u}, \mathbf{p^b}, \mathbf{g}|\mathbf{C}\right) P(\mathbf{C}) = \log P(\mathbf{C}) + \sum_{i=1}^{m} \begin{bmatrix} \log p(b_i|c_i) + \log p(q_i|c_i) + \log p(s_i|c_i) \\ + \log p(p_i^u|c_i) + \log p(p_i^b|c_{i-1}, c_i) \\ + \log p(g_i|t_i) \end{bmatrix} \tag{7}$$

where $c_i$ denotes a character class or a hypothetical category of primitive segment (we call it hyper-category), and $t_i$ denotes SP or NSP. Note that one or more consecutive $c_i$ form a character class $C_j$.

The linguistic prior $P(\mathbf{C})$ is represented by the tri-gram of hyper-categories for primitive segments $P(c_i|c_{i-2}c_{i-1})$. Since the tri-gram of hyper-categories is difficult to obtain, we approximate the tri-gram of hyper-categories $P(c_i|c_{i-2}c_{i-1})$ by that of character classes $P(C_i|C_{i-2}C_{i-1})$, where $C_i$ includes $c_i$:

$$\begin{aligned} \log P(\mathbf{C}) &= \sum_{i=1}^{m} \log P(c_i|c_{i-2}c_{i-1}) \\ &= \sum_{i=1}^{n} \left[ \log P(c_{j_i}|c_{j_i-2}c_{j_i-1}) + \sum_{j=j_i+1}^{j_i+k_i-1} \log P(c_j|c_{j-2}c_{j-1}) \right] \\ &\approx \sum_{i=1}^{n} \left[ \lambda_{11} \log P(C_i|C_{i-2}C_{i-1}) + \lambda_{12} \times \sum_{j=j_i+1}^{j_i+k_i-1} P(C_i|C_{i-2}C_{i-1}) + \lambda_1 \right] \\ &= \sum_{i=1}^{n} \{[\lambda_{11} + \lambda_{12}(k_i - 1)] \times \log P(C_i|C_{i-2}C_{i-1}) + \lambda_1\} \end{aligned} \tag{8}$$

where $\lambda_{11}$ and $\lambda_{12}$ are weighting parameters, and $\lambda_1$ is a bias for balancing the number of characters. We approximate the transition probability of start segment of a character pattern and that of non-start segment using different weights for their varying effects.

Similarly, we approximate the probabilities of the features extracted from primitive segments by the probabilities of those extracted from candidate character patterns and use different weights for the start segment and non-start segment of the other features, obtaining the path score:

$$f(\mathbf{X}, \mathbf{C}) = \sum_{i=1}^{n} \begin{Bmatrix} \sum_{h=1}^{6} [\lambda_{h1} + \lambda_{h2}(k_i - 1)] \log P_h \\ \lambda_{71} \log P(g_{j_i}|SP) + \lambda_{72} \\ \sum_{j=j_i+1}^{j_i+k_i-1} \log P(g_j|NSP) \end{Bmatrix} + n\lambda \tag{9}$$

where $P_h, h = 1, \ldots, 6$, stand for $P(C_i|C_{i-2}C_{i-1})$, $p(b_i|C_i)$, $p(q_i|C_i)$, $p(z_i|C_i)$, $p(p_i^u|C_i)$ and $p(p_i^b|C_{i-1}C_i)$, respectively. $\lambda$ in Eq. (9) embraces all the bias terms for $h = 1, \ldots, 6$. Note that $s_i$ is replaced by $z_i$ here because the probability $p(s_i|c_i)$ is approximated by $p(z_i|C_i)$.

The weighting parameters $\lambda_{h1}, \lambda_{h2} (h = 1 \sim 7)$ and $\lambda$ are selected using a GA to optimize the string recognition performance on a training dataset.

The path score in Eq. (9) is accumulated over the primitive segments, and hence, is insensitive to the number of segmented character patterns. Thus, the optimal path can be found by Viterbi search (dynamic programming).

The path evaluation scorer of Nakagawa et al. [11] and that of Yu et al. [17] can be viewed as a special case of the proposed one in Eq. (9) by setting $\lambda_{h1} = 1, \lambda_{h2} = 0$ (h $= 1 \sim 7$) and $\lambda = 0$ for Nakagawa et al. [11], and by

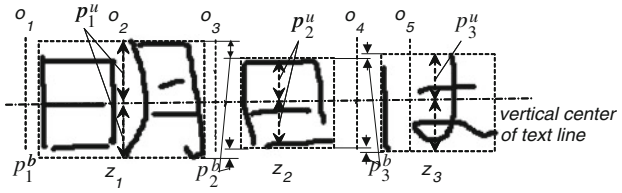



**Fig. 4** Some geometric features

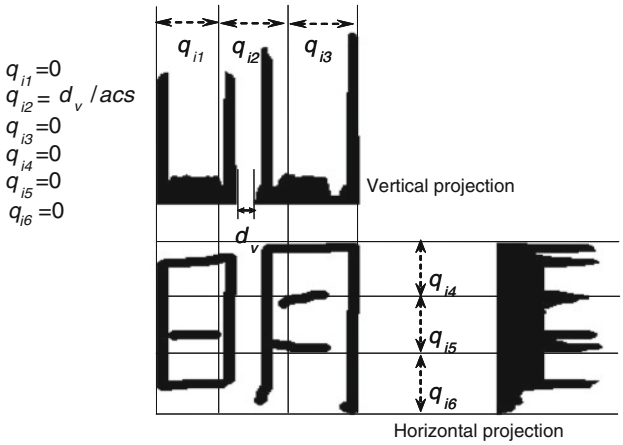



**Fig. 5** Features of character pattern inner gap

setting $\lambda_{41} = \lambda_{42}, \lambda_{h2} = 0$ (h = $1 \sim 3, 5 \sim 7$), and $\lambda = 0$ for Yu et al. [17], respectively.

### 4.2 Evaluation of terms

The tri-gram probability $P(C_i|C_{i-2}, C_{i-1})$ is calculated on a text corpus. It is reduced to unigram or bi-gram when $C_i$ is the first or second character of a sentence. The tri-gram is smoothed to overcome the imprecision of training with insufficient text [19]:

$$P'(C_i|C_{i-2}, C_{i-1}) = \beta_1 P(C_i|C_{i-2}, C_{i-1}) + \beta_2 P(C_i|C_{i-1}) + \beta_3 P(C_i) + \beta_4, \quad (10)$$

where the weights (subject to $\beta_1 + \beta_2 + \beta_3 + \beta_4 = 1$) are obtained by using a different text corpus.

The values of geometric features $b_i, q_i, p_i^u$ and $p_i^b$ are normalized with respect to the average character size *acs* for scaling invariance. Several geometric features are shown in Fig. 4.

The feature vector $b_i$ comprises the height and width of the bounding box of each character pattern.

The feature vector $q_i$ comprises six values as shown in Fig. 5. The first three values represent the horizontal gaps of three vertical slits (partitioned from vertical projection), and the last three ones represent the vertical gaps of three horizontal slits (from horizontal projection).

The feature vector $p_i^u$ comprises the vertical lengths from the center line to the top and bottom of the bounding box.The feature vector $p_i^b$ has two elements measured from the bounding boxes of two adjacent character patterns: the vertical distances between the upper bounds and between the lower bounds. $p(p_1^b|C_1, C_0)$ is set as 1. To reduce the cardinality of $p(p_i^b|C_{i-1}C_i)$, we cluster the character classes into six super-classes according to the mean vector of the unary position features of each class on training samples. $p(p_i^b|C_{i-1}C_i)$ is then replaced by $p(p_i^b|C'_{i-1}, C'_i)$, where $C'_{i-1}, C'_i$ are the super-classes.

The geometric feature vectors $b_i, q_i, p_i^u$ and $p_i^b$ are transformed to log-likelihood scores (to be used in Eq. (9)) using quadratic discriminant function (QDF) classifiers. This is similar to the way that is found in Zhou et al. [15].

The character shape score $p(z_i|C_i)$ is given by a character recognizer, which is detailed in Sect. 6.

The feature vector $g_i$ comprises multiple features measuring the relationship between two primitive segments adjacent to a candidate segmentation point. We approximate $p(\mathrm{g}_i|\mathrm{SP})$ and $p(\mathrm{g}_i|\mathrm{NSP})$ using a SVM classifier. The SVM output is warped to obtain probabilities $p(\mathrm{o}_i|\mathrm{SP})$ and $p(\mathrm{o}_i|\mathrm{NSP})$, where $\mathrm{o}_i$ is the output of the SVM for $g_i$. The warping function is obtained from the distribution of SVM outputs on a validation dataset. $p(\mathrm{o}_1|\mathrm{SP})$ is set as 1.

To warp the SVM outputs, we first obtain the histograms of outputs $p(\mathrm{o}_i|\mathrm{SP})$ and $p(\mathrm{o}_i|\mathrm{NSP})$, then take the cumulative probabilities $p'(\mathrm{o}_i|\mathrm{SP})$ and $p'(\mathrm{o}_i|\mathrm{NSP})$:

$$p'(o_i|SP) = \sum_{l=-\infty}^{O_i} p(l|SP)$$
$$p'(o_i|NSP) = \sum_{l=O_i}^{\infty} p(l|NSP) \quad (11)$$

$p'(\mathrm{o}_i|\mathrm{SP})$ and $p'(\mathrm{o}_i|\mathrm{NSP})$ are then fitted by two sigmoidal functions, with the parameters estimated by minimizing squared errors. This is similar to the method of Platt [20] but uses a different criterion for sigmoidal parameter estimation.

## 5 Parameter optimization

We train the weighting parameters $\lambda_{h1}, \lambda_{h2}$ ($h = 1 \sim 7$) and $\lambda$ by a GA using training data of character string patterns to maximize the recognition rate on training data. To do this, we treat each one of $\lambda_{h1}, \lambda_{h2}$ ($h = 1 \sim 7$) and $\lambda$ as an element of a chromosome. The parameters are estimated by GA in following steps:

(1) Initialization: Initialize $N$ chromosomes with random values from 0 to 1, average fitness of the $N$ chromosomes $f_{\mathrm{old}}$ as 0 and time $t$ as 1.

(2) Crossover: Select two chromosomes at random from $N$ chromosomes. Cross the elements between two random positions to produce two new chromosomes. Repeat until obtaining $M$ new chromosomes.
(3) Mutation: Change each element of $N$+$M$ chromosomes with a random value from $-1$ to 1 at a probability $P_{\text{mut}}$.
(4) Fitness evaluation: Evaluate fitness in terms of the recognition rate on training data with the weight values encoded in each chromosome.
(5) Selection: Decide the roulette probability of each chromosome according to its fitness. First select two chromosomes with the highest fitness, and then select chromosomes using the roulette until obtaining $N$ new chromosomes. Replace the old $N$ chromosomes with the new ones.
(6) Iteration: Obtain the average fitness of the new $N$ chromosomes $f_{\text{new}}$. If $(f_{\text{new}} - f_{\text{old}} < \text{threshold})$ occurs $n_{\text{stop}}$ times or $t > T$, return the chromosome of the highest fitness. Otherwise, set $f_{\text{new}}$ to $f_{\text{old}}$, increment $t$, and go to step 2.

We set $N$ as 50, $M$ as 100, $P_{\text{mut}}$ as 0.03, $n_{\text{stop}}$ as 25 and $T$ as 10,000.

For evaluating the fitness of a chromosome, each training string pattern is searched for the optimal path evaluated using the weight values in the chromosome. To save computation, we first set each weight value as 1 and select the top 100 recognition candidates (segmentation-recognition paths) for each training string. We then train the weight parameters by GA using the selected 100 recognition candidates of each training string pattern. After some iterations, we use the updated weight values to re-select top 100 recognition candidates for each training string pattern. We repeat recognition candidate selection three times.

## 6 Experiments

For evaluating the proposed character string recognition model, we trained the character recognizer and geometric scoring functions using a Japanese on-line handwriting database Nakayosi [21,22]. The character recognizer combines off-line and on-line recognition methods by normalizing the recognition scores to conditional probabilities $p(z_i|C_i)$ [21]. For the geometric scores, four quadratic discriminant function (QDF) classifiers are trained for $p(b_i|C_i)$, $p(q_i|C_i)$, $p(p_i^u|C_i)$ and $p\left(p_i^b|C_{i-1}, C_i\right)$, respectively.

For scoring linguistic context, we prepared an initial tri-gram table from the year 1993 volume of the ASAHI newspaper and the year 2002 volume of the NIKKEI newspaper. We estimated the smoothing parameters $\beta_1, \beta_2, \beta_3, \beta_4$ using the Nakayosi database. The data size of the tri-gram was reduced to 6MB by suppressing non-occurring terms, neglecting a small number of occurrences and quantizing the logarithm values of tri-gram probabilities.

**Table 1** Statistics of training/test text lines

| | # Text lines | # Character patterns | # Character classes | # Characters per line |
|---|---|---|---|---|
| Training | 10,174 | 104,093 | 1,106 | 10.23 |
| Test | 3,511 | 35,686 | 790 | 16.89 |

**Table 2** Results of text line recognition using two over-segmentation schemes

| Performance | Method | |
|---|---|---|
| | Two-stage classification scheme | One-stage classification scheme |
| Test | | |
| $f$ | 0.9934 | 0.9740 |
| $Cr$ (%) | 92.80 | 88.94 |
| $T_{\text{av\_rec\_tl}}$ | 1.32 (s) | 0.99 (s) |

For training the weight parameters and evaluating the performance of character string recognition, we extracted horizontally written text lines from the database Kondate collected from 100 people. We used 75 persons' text lines for training the SVM classifier for the candidate segmentation point probability and the weighting parameters of path evaluation score. After training, we used the text lines of the remaining 25 persons for testing. The statistics of the training and test are listed in Table 1. The experiments were implemented on a Pentium (R) 4 2.80 GHz CPU with 512 MB memory.

First, we compare the performance for over-segmentation by our method (two-stage classification scheme) proposed in this paper and that by direct decision according to the SVM output (one-stage classification scheme) presented in Zhu and Nakagawa [18]. For fair comparison, both methods use the same path evaluation criteria by our model proposed in this paper. The over-segmentation method by one-stage classification extracts 19 features (18 features presented in Zhu and Nakagawa [18] plus the intersecting length feature) from off-strokes and applies the SVM to the extracted features to classify each off-stroke into a segmentation point, a non-segmentation point and an undecided point. We use a character segmentation measure $f$ (F-measure of segmentation point detection), the character recognition rate $C_r$, and average string recognition time $T_{\text{av\_rec\_tl}}$ to evaluate the performance of text line recognition. Table 2 shows the results.

From Table 2, we can see that the over-segmentation method proposed in this paper improves the character recognition and segmentation accuracy remarkably, although it consumes more processing time than the one-stage

classification scheme. The method by two-stage classification scheme leaves many of off-strokes undecided to reduce misclassification, so that it can improve the recognition performance, though undecided segmentation points complicate the candidate lattice and consequently the computation of string recognition.

We also compare the performance of three path evaluation criteria: our model proposed in this paper (Proposed), the one presented in Nakagawa et al. [11] added weighting parameters as shown in Eq. (12) (Method 1) and the one presented in Zhou et al. [15] as shown in Eq. (13) (Method 2). For fair comparison, all the three methods use the same trigram for language context and same classifiers for character recognition and geometric context. The weighting parameters $\lambda_{h1}, \lambda_{h2}$ $(h = 1 \sim 7), \lambda_i$ $(i = 1 \sim 7)$ and $\lambda$ were optimized using the genetic algorithm for each method. The three methods combines the same seven terms in Eq. (9) for path evaluation, but Method 1 does not use the term related to $k_i$ (number of primitive segments composing a character pattern), Method 2 normalizes the path score of Method 1 using the number of segmented characters.

$$f'(\mathbf{X}, \mathbf{C}) = \sum_{i=1}^{n} \begin{pmatrix} \lambda_1 \log P(C_i|C_{i-1}, C_{i-2}) + \lambda_2 \log P(b_i|C_i) \\ +\lambda_3 \log P(q_i|C_i) + \lambda_4 \log P(z_i|C_i) \\ +\lambda_5 \log P(p_i^u|C_i) + \lambda_6 \log P(p_i^b|C_i, C_{i-1}) \\ +\lambda_7 \log P(g_{j_i}|SP) \end{pmatrix} + n\lambda \tag{12}$$

$$f''(\mathbf{X}, \mathbf{C}) = \frac{1}{n} \sum_{i=1}^{n} \begin{pmatrix} \lambda_1 \log P(C_i|C_{i-1}, C_{i-2}) + \lambda_2 \log P(b_i|C_i) \\ +\lambda_3 \log P(q_i|C_i) + \lambda_4 \log P(z_i|C_i) \\ +\lambda_5 \log P(p_i^u|C_i) + \lambda_6 \log P(p_i^b|C_i, C_{i-1}) \\ +\lambda_7 \log P(g_{j_i}|SP) \end{pmatrix} \tag{13}$$

For Method 2, we use beam search for finding the optimal paths in the candidate lattice, because the path score is not cumulative with the character sequence. For the proposed method and the Method 1, the optimal paths are found by Viterbi search. For all the three methods, the candidate lattice retains 10 candidate classes for each character pattern.

To justify weighting parameter optimization by GA, we also draw a comparison between the proposed character recognition rate optimization by GA (CR-GA) and the minimum classification error (MCE) criterion [23] optimized by stochastic gradient decent [24] (MCE-SGD). MCE-SGD is to find the optimal parameter vector $\lambda$ by minimizing the following difference between the scores of the most confusing string class and that of the correct one:

$$\begin{aligned} &L_{\mathrm{MCE}}(\boldsymbol{\lambda}, \mathbf{X}) = \sigma(\max(Score_{\mathrm{Incorrect}}) - Score_{\mathrm{Correct}}) \\ &\sigma(x) = (1 + e^{-x})^{-1} \\ &Score_{\mathrm{correct}} = \text{score of the correct path} \\ &\qquad\qquad \text{in the candidate lattice} \\ &Score_{\mathrm{incorrect}} = \text{scores of incorrect paths} \\ &\qquad\qquad \text{in the candidate lattice} \end{aligned} \tag{14}$$

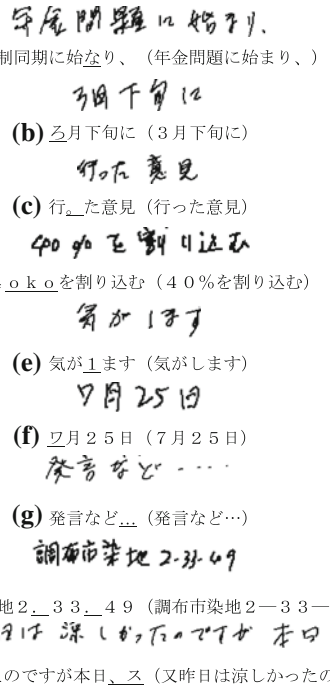
**Table 3** Results of text line recognition by three path evaluation methods

| Performance | Method | | |
|---|---|---|---|
| | Proposed | Method 1 | Method 2 |
| CR-GA | | | |
| Training | | | |
| $f$ | 0.9941 | 0.9906 | 0.9850 |
| $Cr$ (%) | 92.65 | 91.68 | 91.12 |
| $T_{\mathrm{av\_rec\_tl}}$ | 1.31 (s) | 1.32 (s) | 1.32 (s) |
| Test | | | |
| $f$ | 0.9934 | 0.9903 | 0.9827 |
| $Cr$ (%) | 92.80 | 91.94 | 91.07 |
| $T_{\mathrm{av\_rec\_tl}}$ | 1.32(s) | 1.33 (s) | 1.33 (s) |
| MCE-SGD | | | |
| Training | | | |
| $f$ | 0.9940 | 0.9910 | 0.9855 |
| $Cr$ (%) | 92.25 | 91.57 | 90.89 |
| $T_{\mathrm{av\_rec\_tl}}$ | 1.31 (s) | 1.32 (s) | 1.32 (s) |
| Test | | | |
| $f$ | 0.9937 | 0.9905 | 0.9824 |
| $Cr$ (%) | 92.34 | 91.77 | 90.66 |
| $T_{\mathrm{av\_rec\_tl}}$ | 1.33 (s) | 1.33 (s) | 1.33 (s) |

Table 3 shows the string recognition results of the three path evaluation methods. For reference, the trained weight values of Eq. (9) by CR-GA are as follows:
$(\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22}, \lambda_{31}, \lambda_{32}, \lambda_{41}, \lambda_{42}, \lambda_{51}, \lambda_{52}, \lambda_{61}, \lambda_{62}, \lambda_{71}, \lambda_{72}, \lambda) = (0.351, 0.000, 0.265, 0.001, 0.199, 0.000, 1.000, 0.641, 0.009, 0.000, 0.100, 0.000, 0.323, 0.120, 0.100)$.

From the weighting parameters obtained by GA, we can see that except the character recognition score $p(z_i|C_i)$ and the non-segmentation point score $p(g_i|\mathrm{NSP})$, the other geometric features and linguistic context are not weighted with the number of primitive segments ($\lambda_{h2} = 0$). This implies that except the character recognition score and the non-segmentation point score, the other geometric features and linguistic context are almost independent of the number of primitive segments of character pattern.

From the results, we can see that either by CR-GA or by MCE-SGD, our proposed path evaluation model improves



**(a)** 年制同期に始なり、（年金問題に始まり、）

**(b)** 乙月下旬に（３月下旬に）

**(c)** 行、た意見（行った意見）

**(d)** ４ｏｋｏを割り込む（４０％を割り込む）

**(e)** 気が１ます（気がします）

**(f)** 刀月２５日（７月２５日）

**(g)** 発言など…（発言など…）

**(h)** 調布市染地２．３３．４９（調布市染地２－３３－４９）

**(i)** ス昨日は涼しかったのですが本日、ス（又昨日は涼しかったのですが本日は）

**Fig. 6** Examples of recognition errors. The text below each string pattern is the recognition result, followed by ground-truth. **i** It has a segmentation error

the character recognition and segmentation accuracy. The Method 1 tends to over-merge characters because a shorter character sequence tends to have larger evaluation score than a longer one. On the other hand, the Method 2 (normalized path score) is biased to longer strings, and so, tends to over-split characters. Our proposed model overcomes these problems, and the path score is insensitive to the number of segmented characters. The three methods consume nearly the same processing time. The parameter optimization method CR-GA yields better string recognition performance than the MCE-SGD. This can be attributed to the local optimum of gradient descent for MCE-SGD. The CR-GA directly optimizes the character recognition rate (which is not differentiable) on training data and achieves a global optimum.

Figure 6 shows some examples of misrecognition and mis-segmentation given by the proposed model. For each example, the upper line is the written text, and the lower line is the recognition result followed by the correct result (ground-truth) where the recognition errors are highlighted by underlines. We observed two major sources causing segmentation-recognition errors.

(1) Problem of character recognition: Fig. 6a–e show recognition errors due to character recognition, where the correct answers are not within the top 10 candidate classes output by the character recognizer for each character pattern. To solve this, we need to improve the character recognition accuracy. Increasing the number of candidate classes can reduce the missing of correct class, but seriously complicates the search space of candidate lattice.

(2) Problems of path evaluation and over-segmentation: Fig. 6 f–i show recognition errors due to path evaluation and over-segmentation. Correct character answers are within the top 10 candidate classes but the path evaluation fails to find the correct one. To solve this, we should improve the accuracy of the linguistic context score and the geometric features scores, and that of over-segmentation.

## 7 Conclusion

In this paper, we presented a robust on-line handwritten Japanese character string recognition model that can evaluate the likelihood of candidate segmentation and its corresponding string class by combining the scores of character recognition, geometric and linguistic contexts. With the path evaluation criterion balanced by the primitive segment number for the scores associated with the candidate character patterns on the path, the proposed text recognition model can effectively overcome the variable length of candidate segmentation. With the model parameters optimized by GA, the proposed system outperforms the other popular path evaluation criteria in our experiments. The optimized weighting parameters justify the fact that only the character recognition score and the non-segmentation point score are dependent on the primitive segmentation number of candidate character patterns.

To further improve the segmentation and recognition performance is the aim of our future work. This can be achieved by incorporating more effective geometric features, exploiting better geometric context likelihood functions and weighting parameter learning methods and improving the accuracy of character recognizer. To speed up recognition is another dimension of our future work. We should consider effective methods to remove invalid patterns from the lattice.

**Acknowledgments** This work was supported in part by Grant-in-Aid for Scientific Research under contract no. (B)17300031, the R&D fund for “development of pen & paper based user interaction” under Japan Science and Technology Agency, and the Natural Science Foundation of China (NSFC) under contract no.60775004.

## Appendix

Geometric features of SVM classifier on over-segmentation in [18]

First, we define the following terminology:

| | |
|---|---|
| $Bb_p$: | Bounding box of the immediately preceding stroke |
| $Bb_s$: | Bounding box of the immediately succeeding stroke |
| $BB_{p\_all}$: | Bounding box of all the preceding strokes |
| $BB_{s\_all}$: | Bounding box of all the succeeding strokes |
| *acs:* | Average character size |
| $DB_x$: | Distance between $BB_{p\_all}$ and $BB_{s\_all}$ to $x$-axis<br>$DB_x$ = X coordinate of the left position of $BB_{s\_all}$-X coordinate of the right position of $BB_{p\_all}$ |
| $DB_y$: | Distance between $BB_{p\_all}$ and $BB_{s\_all}$ to $y$-axis<br>$DB_y$ = Y coordinate of the top position of $BB_{s\_all}$-Y coordinate of the bottom position of $BB_{p\_all}$ |
| $D_{bx}$: | Distance between $Bb_p$ and $Bb_s$ to $x$-axis<br>$D_{bx}$ = X coordinate of the left position of $Bb_s$-X coordinate of the right position of $Bb_p$ |
| $D_{by}$: | Distance between $Bb_p$ and $Bb_s$ to $y$-axis<br>$D_{by}$ = Y coordinate of the top position of $Bb_s$-Y coordinate of the bottom position of $Bb_p$ |
| $O_b$: | Overlap area between $Bb_p$ and $Bb_s$ |
| $D_{bsy}$: | Distance between centers of $Bb_p$ and $Bb_s$ to $y$-axis<br>$D_{bsy}$ = Y coordinate of the center of $Bb_s$-Y coordinate of the center of $Bb_p$ |
| $D_{bs}$: | Absolute distance of centers of $Bb_p$ and $Bb_s$ |
| $Df_b$: | Difference between $BB_{p\_all}$ and $Bb_s$<br>$Df_b$ = abs(Y coordinate of the top position of $BB_{p\_all}$-Y coordinate of the top position of $Bb_s$) |

Then, the following 18 features of off-strokes are extracted for over-segmentation:

| | |
|---|---|
| $f_1$: | Passing time for the off stroke |
| $f_2$: | $DB_x$ / acs |
| $f_3$: | Overlap area between $BB_{p\_all}$ and $BB_{s\_all}$ / $(\text{acs})^2$ |
| $f_4$: | $D_{bx}$ / width of $Bb_p$ |
| $f_5$: | $D_{bx}$ / width of $Bb_s$ |
| $f_6$: | $D_{bx}$ / acs |
| $f_7$: | $D_{by}$ / height of $Bb_p$ |
| $f_8$: | $D_{by}$ / height of $Bb_s$ |
| $f_9$: | $D_{by}$ / acs |
| $f_{10}$: | $O_b$ / (width x height of $Bb_s$) |
| $f_{11}$: | $O_b$ / $(\text{acs})^2$ |
| $f_{12}$: | $D_{bsy}$ / acs |
| $f_{13}$: | $D_{bs}$ / acs |
| $f_{14}$: | $Df_b$ / acs |
| $f_{15}$: | Length of the off-stroke / acs |
| $f_{16}$: | Sine value of the off-stroke |
| $f_{17}$: | Cosine value of the off-stroke |
| $f_{18}$: | $f_2$ / the maxinum $f_2$in text |

## References


1. Tseng, L.Y., Chen, R.C.: Segmenting handwritten Chinese characters based on heuristic merging of stroke bounding boxes and dynamic programming. Pattern Recognit. Lett. **19**(10), 963–973 (1998)
2. Zhao, S., Chi, Z., Shi, P., Yan, H.: Two-stage segmentation of unconstrained handwritten Chinese characters. Pattern Recognit. **36**(1), 145–156 (2003)
3. Liang, Z., Shi, P.: A metasynthetic approach for segmenting handwritten Chinese character strings. Pattern Recognit. Lett. **26**(10), 1498–1511 (2005)
4. Furukawa, N., Tokuno, J., Ikeda, H.: Online character segmentation method for unconstrained handwriting strings using off-stroke features. In: Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition, pp. 361–366. La Baule, France (2006)
5. Liu, C.-L., Sako, H., Fujisawa, H.: Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1395–1407 (2004)
6. Cheriet, M., Kharma, N., Liu, C.-L., Suen, C.Y.: Character Recognition Systems: A Guide for Students and Practitioners. Wiley, New York (2007)
7. Su, T.-H., Zhang, T.-W., Guan, D.-J., Huang, H.-J.: Off-line recognition of realistic Chinese handwriting using segmentation-free strategy. Pattern Recognit. **42**(1), 167–182 (2008)
8. Murase, H.: Online recognition of free-format Japanese handwritings. In: Proceedings of the 9th International Conference on Pattern Recognition, vol. 2, pp. 1143–1147. Rome, Italy (1988)
9. Fukushima, T., Nakagawa, M.: On-line writing-box-free recognition of handwritten Japanese text considering character size variations. In: Proceedings of the 15th International Conference on Pattern Recognition, vol. 2, pp. 359–363. Barcelona, Spain (2000)
10. Senda, S., Yamada, K.: A maximum-likelihood approach to segmentation-based recognition of unconstrained handwriting text. In: Proceedings of the 6th International Conference on Document Analysis and Recognition, pp. 184–188. Seattle, WA (2001)
11. Nakagawa, M., Zhu, B., Onuma, M.: A model of on-line handwritten Japanese text recognition free from line direction and writing format constraints. IEICE Trans. Inf. Syst. E **88**(D(8), 1815–1822 (2005)
12. Ding, X., Liu, H.: Segmentation-driven offline handwritten Chinese and Arabic script recognition. In: Proceedings of the Summit on Arabic and Chinese Handwriting, pp. 61–73. College Park, USA (2006)
13. Zhu, B., Nakagawa, M.: On-line handwritten Japanese text recognition by improving segmentation quality. In: Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition, pp. 379–384. Montréal, Canada (2008)
14. Tulyakov, S., Govindaraju, V.: Probabilistic model for segmentation based word recognition with lexicon. In: Proceedings of the 6th International Conference on Document Analysis and Recognition, pp. 164–167. Seattle, WA (2001)
15. Zhou, X.-D., Yu, J.-L., Liu, C.-L., Nagasaki, T., Marukawa, K.: Online handwritten Japanese character string recognition incorporating geometric context, In: Proceedings of the 9th International Conference on Document Analysis and Recognition, pp. 48–52. Curitiba, Brazil (2007)

16. Chen, M.Y., Kundu, A., Srihari, S.N.: Variable duration hidden Markov model and morphological segmentation for handwritten word recognition. IEEE Trans. Image Process. **4**(12), 1675–1687 (1995)
17. Yu, J.-L., Zhou, X.-D., Liu, C.-L.: Search strategies in online handwritten character string recognition (in Chinese). In: Proceedings of the 1st Chinese Conference on Pattern Recognition, pp. 299–305. Beijing, China (2007)
18. Zhu, B., Nakagawa, M.: Segmentation of on-line freely written Japanese text using SVM for improving text recognition. IEICE Trans. Inf. Syst. E **E91**(D1), 105–113 (2008)
19. Mori, H., Aso, H.: Postprocessing for Japanese document recognition based and 2nd order Markov model (in Japanese). Technical report of IPS J, NL-94-63, pp. 89–96. (1994)
20. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola, A.J., Bartlett, P., Scholkopf, D., Schuurmanns, D. (eds.) Advances in Large Margin Classifiers, MIT Press, Cambridge (1999)
21. Oda, H., Zhu, B., Tokuno, J., Onuma, M., Kitadai, A., Nakagawa, M.: A compact on-line and off-line combined recognizer. In: Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition, pp. 133–138. La Baule, France (2006)
22. Nakagawa, M., Matsumoto, K.: Collection of on-line handwritten Japanese character pattern databases and their analysis. Int. J. Document Anal. Recognit. **7**(1), 69–81 (2004)
23. Juang, B.-H., Katagiri, S.: Discriminative learning for minimum error classification. IEEE Trans. Signal Process. **40**(12), 3043–3054 (1992)
24. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)