IEICE TRANS. INF. & SYST., VOL.E99–D, NO.4 APRIL 2016

PAPER

Character-Position-Free On-Line Handwritten Japanese Text Recognition by Two Segmentation Methods*

Jianjuan LIANG[†a)], ***Student Member***, **Bilan ZHU**[†], ***Member***, **Taro KUMAGAI**[††], ***Nonmember***, ***and*** **Masaki NAKAGAWA**[†b)], ***Fellow***

SUMMARY The paper presents a recognition method of character-position-free on-line handwritten Japanese text patterns to allow a user to overlay characters freely without confirming previously written characters. To develop this method, we first collected text patterns written without wrist or elbow support and without visual feedback and then prepared large sets of character-position-free handwritten Japanese text patterns artificially from normally handwritten text patterns. The proposed method sets each off-stroke between real strokes as undecided and evaluates the segmentation probability by SVM model. Then, the optimal segmentation-recognition path can be effectively found by Viterbi search in the candidate lattice, combining the scores of character recognition, geometric features, linguistic context, as well as the segmentation scores by SVM classification. We test this method on variously overlaid sample patterns, as well as on the above-mentioned collected handwritten patterns, and verify that its recognition rates match those of the latest recognizer for normally handwritten horizontal Japanese text with no serious speed restriction in practical applications.

key words: *character-position-free, on-line recognition, handwritten text recognition, candidate lattice, SVM classification*

1. Introduction

On-line handwritten text recognition has been receiving large attention, especially for unconstrained text recognition due to the development and growing popularity of pen-based or touch-based input devices, such as smart phones, tablet PCs and electronic whiteboards [1].

This trend is spreading into automobiles where drivers may input destinations to navigation systems by speech or written characters. In this situation, however, they often have to write characters without supported by wrist or elbow and without visual feedback. Moreover, they may stop writing while writing destinations to keep safe driving, namely, a text may be written in one or more steps, so that positions of characters or even strokes (a trajectory written from pen/finger down to up) become unstable. In this paper, we propose a method for character-position-free on-line handwritten Japanese text recognition.

The research on on-line handwriting recognition started in the 1960s and has been receiving intensive interest from the 1980s. Tappert et al. [2] has given a comprehensive survey before the 1990s. In recent survey papers, Liu et al. [3] reviewed the advances in on-line Chinese and Japanese handwriting recognition from the 1990s, and mentioned the integration of segmentation and recognition method has particular importance for handwritten text recognition. Recently, Zhu et al. [4] reviewed the on-line handwriting Japanese character recognition and its practical applications. The framework to solve the problem seems to have been made, but it is rather recent that components in the framework have been matured to the level of practical use. Moreover, new needs are coming from special but potentially important applications, such as recognition of overlaid handwritten characters, characters written without visual feedback and so on. The existing systems cannot recognize these handwriting text patterns, although the recognition framework has been established and various machine learning methods have been developed. There are several attempts being made to solve the problems as we will review several methods below.

On-line handwritten text recognition is more challenging than isolated character recognition due to the added ambiguity of character segmentation. An early work was presented by Murase et al. [5] for Japanese text recognition. It generates a candidate character lattice based on the segmentation threshold, and searches for an optimal character string using the scores of character recognition. Recently, Zhu et al. [6] proposed an over-segmentation based string recognition method. Firstly, a handwritten string pattern is over-segmented into primitive segments according to the geometric features such as distance and overlap between adjacent strokes. The primitive segments are combined to generate candidate character patterns. Each candidate character pattern is associated with a number of candidate classes with confidence scores. Then, a segmentation-recognition candidate lattice is constructed based on the combination of all candidate character patterns and character classes. Finally, an optimal path is searched from this lattice by the Viterbi algorithm combing the scores of character recognition, geometric features, linguistic context and segmentation scores. This recognition method has been successfully applied in smart phones and tablets.

On the other hand, there have been some works on the overlaid handwritten text segmentation and recognition which recognizes characters written overlaid in the same

Manuscript received November 27, 2015.
Manuscript publicized January 6, 2016.
[†]The authors are with Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology, Koganei-shi, 184–8588 Japan.
[††]The author is with Automotive Electronics Development Center, Mitsubishi Electric Corporation, Sanda-shi, 669–1513 Japan.
*This is a paper on system development.
a) E-mail: jianjuanliang@gmail.com
b) E-mail: nakagawa@cc.tuat.ac.jp (Corresponding author)
DOI: 10.1587/transinf.2015EDP7479

small area of handheld devices such as mobile phones. Shimodaira et al. [7] introduced substroke Hidden Markov Models (HMMs) with a bigram language model for overlaid handwritten Japanese text. Using a bigram model consisting of 1,016 Japanese educational Kanji and 71 Hiragana characters, the character recognition rates are 74.9% for free stroke order patterns and 91.1% for fixed stroke order patterns. Tonouchi et al. [8] proposed an on-line overlaid handwriting recognition system based on stroke-level discrete Markov Models to recognize 81 hiragana characters and 5 symbols. The recognized hiragana characters can be easily translated into kanji characters by gestures.

In overlaid English handwriting recognition, Bharath A. et al. [9] proposed a HMM-based recognition system for overwritten words. The word recognition rate is 89% with a 20K word lexicon.

To input overlaid Chinese handwriting, Zou et al. [10] proposed a quick segmentation method based on an artificial neural network to detect the first stroke of each character, and then let previous characters fade out for the clear viewing of the current character. The quick segmentation method is also used in the recognition system to speed up the whole recognition process. Overlaid handwritten text is recognized by combining scores of character recognition and bigram scores. Wan et al. [11] proposed a method of combining stroke level evaluation by SVM and character level evaluation based on character recognition scores, bigram scores and geometric scores and then proposed a strategy to filter out correct segmentations. They report that the strategy performs better than the DP algorithm. Lv et al. [12] proposed a real-time overlaid handwriting recognition method. It firstly over-segments a stroke sequence based on SVM into primitive segments, which may be concatenated into candidate characters during the next path search. Then, it searches for a best path by integrating class-independent unary and binary geometric scores, character classification scores and linguistic context.

In this paper, we investigate handwritten text patterns written without supported by wrist or elbow and without visual feedback, and make models to produce such text patterns from normally handwritten text patterns as well as the model for text patterns with characters completely overlaid. Then, we consider recognition methods which can recognize handwritten text patterns produced from all the models. The proposed method has been employed for the EMIRAI 3 xDAS assisted-driving concept car, which was exhibited during the 44th Tokyo Motor Show 2015 [20].

The rest of this paper is organized as follows: Sect. 2 presents sample pattern datasets. Section 3 describes an overview of our recognition methods. Section 4 shows our experiments. Finally, Sect. 5 draws our concluding remarks.

2. Character-Position-Free On-Line Handwritten Text

Since we do not have actual on-line handwritten text patterns collected from vehicle-mounted touch panels while driving, we have collected a small amount of text patterns from our laboratory students with the condition that they write specified short phrases on a tablet PC without supported by wrist or elbow and without confirming previously written strokes, namely, without visual feedback for the safety of driving, which meets the requirement for display system operation while a vehicle is in motion [19]. This is a simulated environment where a driver would write destinations under such the condition. Figure 1 shows examples of collected text patterns. Characters are often partially overlapped on each other or separated randomly while stroke positions are relatively stable within a character pattern although displaced strokes make character patterns sometimes difficult to read.

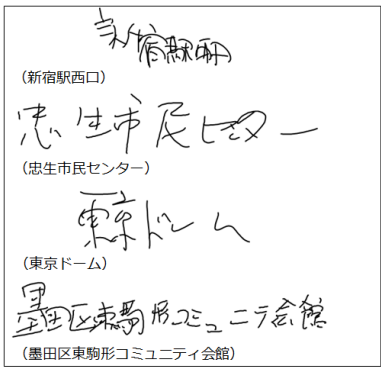


Fig. 1 Examples of collected handwritten text patterns.

JAMA (Japan Automobile Manufacturers Association, Inc.) publishes the guidelines for in-vehicle display systems [19], where the following requirements are related to this study: the operation of a display system shall not result in a marked obstruction of forward field visibility; information to be presented by a display system shall not cause the driver to gaze at the screen continuously. According to the requirements, we allow a driver to write text without visual feedback.

In order to develop a recognizer which can recognize on-line Japanese text patterns written without physical support and visual feedback, we make models to produce such text patterns from normally handwritten text patterns in the Kondate database [16] as well as the model for text patterns with characters completely overlaid.

We make 4 models and generate 4 datasets by changing the parameters. We call them character-position-free handwritten text models and character-position-free handwritten text datasets, respectively. When we make the datasets, we exclude all punctuation marks following the previous research [12] since their recognition is difficult as independent symbols but they can be easily input by soft keys or gestures.

Model 1 simulates handwritten text patterns where a character is placed randomly from a half character-size to a full-size advanced from left to right with 10% variations vertically according to Eq. (1) where d_x and d_y stand for the horizontal distance and the vertical distance from the previous character to the next character, respectively, $\overline{x}$ and $\overline{y}$ are the average width and the average height of characters in

handwritten text, respectively. Generated handwritten text patterns are stored in Dataset 1.

For the random number generation, we compared the uniform random and the normal random. The uniform random generator generates text patterns more similar to the collected patterns. Therefore, we choose it to generate the random distance for the other models.

$$d_x \sim U([0.5 * \overline{x}, 1.0 * \overline{x}])$$
$$d_y \sim U([-0.1 * \overline{y}, 0.1 * \overline{y}]) \quad (1)$$

Model 2 simulates handwritten text patterns where a character is placed randomly from 0.4 times to 1.5 times the character-size advanced from left to right with 10% variations vertically according to Eq. (2). It has a wider variation horizontally compared with Model 1. Generated handwritten text patterns are stored in Dataset 2.

$$d_x \sim U([0.4 * \overline{x}, 1.5 * \overline{x}])$$
$$d_y \sim U([-0.1 * \overline{y}, 0.1 * \overline{y}]) \quad (2)$$

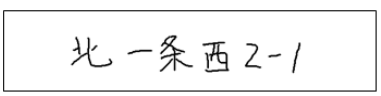
Model 3 simulates overlaid handwritten text patterns where characters are overlaid on previous characters according to Eq. (3). Generated overlaid handwritten text patterns are stored in Dataset 3.

$$d_x \sim U([-0.1 * \overline{x}, 0.1 * \overline{x}])$$
$$d_y \sim U([-0.1 * \overline{y}, 0.1 * \overline{y}]) \quad (3)$$

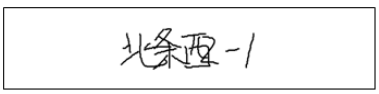
Model 4 simulates handwritten text patterns where a character is placed randomly in any direction (left, right, top, bottom etc.) of the immediately preceding character according to Eq. (4). Generated handwritten text patterns are stored in Dataset 4.

$$d_x \sim U([-1.0 * \overline{x}, 1.0 * \overline{x}])$$
$$d_y \sim U([-1.0 * \overline{y}, 1.0 * \overline{y}]) \quad (4)$$

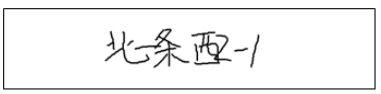
Figure 2 shows an original handwritten text pattern and generated patterns by Model 1 to Model 4.



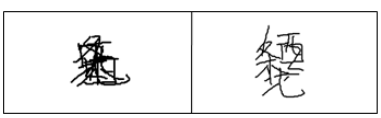
(a) An original handwritten text pattern.



(b) Generated Dataset 1.



(c) Generated Dataset 2.



(d) Generated Dataset 3(Left) and Dataset 4(Right).

Fig. 2 Examples of generated character-position-free handwritten text patterns.

3. Character-Position-Free On-Line Handwritten Text Recognition

The character-position-free on-line handwritten text recognition method has three major steps: over-segmentation, candidate lattice construction and handwritten text recognition by optimal path search, as shown in Fig. 3. Each step is described in detail in the following subsections.

3.1 Over-Segmentation

A handwritten text pattern is composed of many characters with a sequence of strokes, In Japanese, different kinds and complexities of characters: Kanji, Hiragana, Katakana, numeric characters and the others are mixed. An input text pattern should be correctly segmented into each character as far as possible. It is difficult, however, due to the facts that spaces between characters are not obvious, many characters include multiple radicals with internal gaps and some characters are connected in writing. To solve these problems, a text pattern is over-segmented into a sequence of primitive segments so as to segment true segmentation points surely but may segment single character patterns into pieces, which could be combined in the later text recognition stage. Zhu et al. employ two-stage segmentation scheme [6]. In the first stage, each off-stroke (a vector from the last point of a previous stroke to the first point of the next stroke) is classified into non-segmentation point (NSP) and hypothetical one based on geometric features. Then, in the second stage, each hypothetical point is classified into segmentation point (SP) and undecided point (UP) using SVM model according to 20-dimensional features extracted from an off-stroke, where a SP separates two characters at the off-stroke, an NSP indicates the off-stroke is within a character and a UP is interpreted either as a SP or an NSP. When UP is interpreted as a SP, it is used to extract candidate character patterns beside it with nearest neighbor SPs or UPs interpreted as SPs. When UP is interpreted as an NSP, it is considered within a

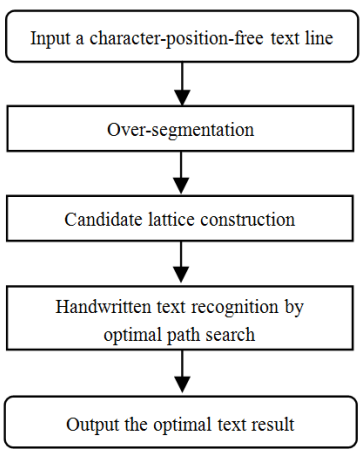


Fig. 3 Flow chart of recognition process.

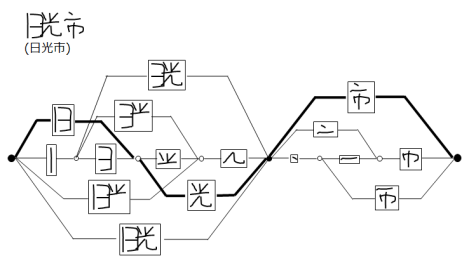
character pattern and does not play a role for segmentation. We call a sequence of strokes delimited by SP or UP as a primitive segment.

In a character-position-free handwritten text pattern, however, spaces between characters are very unstable. We can't directly use the conventional handwritten text recognition model. The first stage in the above-mentioned recognizer may combine two characters since the space between them disappears. Therefore, we remove the first stage and only employ the second stage.

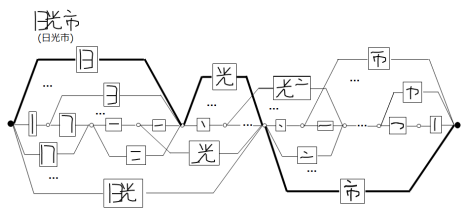
The next concern is the classification of off-strokes into NSP, SP or UP. We may follow this scheme or change the scheme. In this paper, we compare two segmentation methods. The first one is the conventional method to classify off-strokes into NSP, SP or UP although all the parameters and thresholds are retrained according to the new training patterns. We call this method "candidate segmentation method (CSM)". On the other hand, we set every off-stroke as UP in the alternative method although we employ the output of SVM model in the text recognition stage. We call it "undecided segmentation method (USM)".

Namely, the first method classifies off-strokes into NSP, SP or UP, but the second method treats every off-stroke as UP. Both of the two methods, however, transform the output of SVM to segmentation probability value. Moreover, the segmentation probability value is combined into the optimal path evaluation in candidate segmentation-recognition paths.

Figure 4(a) shows segmentation by CSM and Fig. 4(b) shows that by USM, respectively, where a node is a SP and a rectangle is a candidate character pattern. Rectangles between adjacent SPs are primitive segments. A segmentation path connects candidate character patterns following SPs from the start to the end. The thickly marked path is the correct segmentation sequence. We delete candidate character patterns if their widths are longer than the threshold. It is clear that USM has more SPs than CSM.



(a) Segmentation by the candidate segmentation method.



(b) Segmentation by the undecided segmentation method.

Fig. 4 Over-segmentation.

In both the methods, we need to extract more geometric features from an off-stroke in order to enhance the reliability of over-segmentation. Through investigation into related literatures, we employ all the useful geometric features proposed so far, i.e., 56-dimensional features. The detail will be described in the next subsection.

3.2 Features for SVM

An off-stroke is evaluated by SVM. We cover all the useful geometric features from the literature [12] and [13], and extract 56 features from each off-stroke. Table 1 shows the features and Table 2 shows terms to derive the 56 features. Most of the features are normalized by an average character size (acs). The average character size is estimated by measuring the length of the longer side of the bounding box for each stroke, sorting the lengths from all the strokes and taking the average of the larger 1/3 of them.

We just use order information for the on-line character recognition engine, but we do not use time information. Time information can be used to separate characters intentionally, but it causes mis-segmentation when a user stops writing in a character pattern. In automobile environment and even in ordinary environment, a user may stop writing or resume writing.

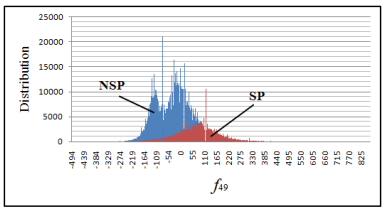
We examined the distribution of each feature using all the training patterns. We have found 20 features are clearly essential such as f_{49} as shown in Fig. 5(a) where the two classes (NSP and SP) are divided to some extent, while other features such as f_{21} cannot divide the two classes as shown

Table 1 Features extracted from an off-stroke.

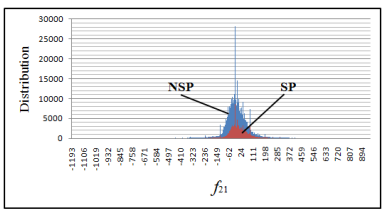
No.	Definition	No.	Definition
f_1	DB_x/acs	f_{29}	D_{rl}/acs
f_2	$O_{all}/(acs)^2$	f_{30}	width of S_p/acs
f_3	D_{bx}/width of B_p	f_{31}	height of S_p/acs
f_4	D_{bx}/width of B_s	f_{32}	width of S_s/acs
f_5	D_{bx}/acs	f_{33}	height of S_s/acs
f_6	D_{by}/height of B_p	f_{34}	log(width/height of S_p)
f_7	D_{by}/ height of B_s	f_{35}	log(width/height of S_s)
f_8	D_{by}/acs	f_{36}	L_p/acs
f_9	O_b/area of B_p	f_{37}	L_s/acs
f_{10}	O_b/area of B_s	f_{38}	square root of B_p/acs
f_{11}	O_b/acs^2	f_{39}	square root of B_s/acs
f_{12}	D_{bsx}/acs	f_{40}	x-coordinate of P_{1e}/acs
f_{13}	D_{bsy}/acs	f_{41}	y-coordinate of P_{1e}/acs
f_{14}	D_{bs}/acs	f_{42}	x-coordinate of P_{2s}/acs
f_{15}	Df_b/acs	f_{43}	y-coordinate of P_{2s}/acs
f_{16}	L_{off}/acs	f_{44}	DP_x/acs
f_{17}	$\sin(L_{off})$	f_{45}	DP_y/acs
f_{18}	$\cos(L_{off})$	f_{46}	D_{1elp}/acs
f_{19}	$f_1/\max(f_1)$ in text	f_{47}	D_{1ebp}/acs
f_{20}	x-center of S_p/acs	f_{48}	D_{2srp}/acs
f_{21}	y-center of S_p/acs	f_{49}	D_{2sbp}/acs
f_{22}	x-center of S_s/acs	f_{50}	D_{2els}/acs
f_{23}	y-center of S_s/acs	f_{51}	D_{2ebs}/acs
f_{24}	D_l/acs	f_{52}	width of S_{ps}/acs
f_{25}	D_r/acs	f_{53}	height of S_{ps}/acs
f_{26}	D_t/acs	f_{54}	log(width/height of S_{ps})
f_{27}	D_b/acs	f_{55}	x-center of S_{ps}/acs
f_{28}	D_{bt}/acs	f_{56}	y-center of S_{ps}/acs

Table 2 Terms to derive features.

Term.	Description
acs	Average character size of text line
B_p	Bounding box of immediately preceding stroke
B_s	Bounding box of immediately succeeding stroke
B_{p_all}	Bounding box of all preceding strokes
B_{s_all}	Bounding box of all succeeding strokes
DB_x	Distance between B_{p_all} and B_{s_all} in x-axis
D_{bx}	Distance between B_p and B_s in x-axis
D_{by}	Distance between B_p and B_s in y-axis
O_b	Overlap area between B_p and B_s
O_{all}	Overlap area between B_{p_all} and B_{s_all}
D_{bsx}	Distance between centers of B_p and B_s in x-axis
D_{bsy}	Distance between centers of B_p and B_s in y-axis
D_{bs}	Absolute distance of centers of B_p and B_s
Df_b	Vertical distance between B_{p_all} and B_s
L_{off}	Length of off-stroke
S_p	Immediately preceding stroke
S_s	Immediately succeeding stroke
S_{ps}	Union of S_p and S_s
P_{1e}	End point of S_p
P_{2s}	Start point of S_s
P_{2e}	End point of S_s
L_p	Length of S_p
L_s	Length of S_s
D_l	Distance between left bounds of B_p and B_s
D_r	Distance between right bounds of B_p and B_s
D_t	Distance between top bounds of B_p and B_s
D_b	Distance between bottom bounds of B_p and B_s
D_{bt}	Distance between bottom-top bounds of B_p and B_s
D_{rl}	Distance between right-left bounds of B_p and B_s
DP_x	Distance between P_{1e} and P_{2s} in x-axis
DP_y	Distance between P_{1e} and P_{2s} in y-axis
D_{1elp}	Distance between P_{1e} and left bound of B_p
D_{1ebp}	Distance between P_{1e} and bottom bound of B_p
D_{2srp}	Distance between P_{2s} and right bound of B_p
D_{2sbp}	Distance between P_{2s} and bottom bound of B_p
D_{2els}	Distance between P_{2e} and left bound of B_s
D_{2ebs}	Distance between P_{2e} and bottom bound of B_s



(a) Distribution of f_{49}.



(b) Distribution of f_{21}.

Fig. 5 Distributions of f_{49} and f_{21} features in training patterns.

in Fig. 5(b). Table 3 shows the 20 essential features. We still keep using all 56 features, however, so that they contribute to the segmentation of character-position-free on-line handwritten Japanese text using SVM.

We train the SVM model using training patterns of off-strokes with setting SP to 1 and NSP to -1.

SVM actually classify each off-stroke into NSP, SP or UP for CSM, but it does not for USM. For both of them, however, it produces probability of an off-stroke as NSP or SP.

Table 3 Essential features.

Essential features	Number
f_1, f_2, f_5, f_8, f_{12}, f_{13}, f_{14}, f_{16}, f_{24}, f_{25}, f_{26}, f_{27}, f_{28}, f_{29}, f_{44}, f_{45}, f_{48}, f_{49}, f_{52}, f_{53}	20

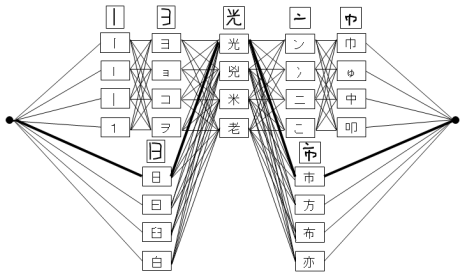


Fig. 6 Candidate lattice with two segmentation paths.

3.3 Candidate Lattice Construction

Each candidate character pattern is associated with a number of candidate classes with confidence scores from character classification. The combination of all candidate character patterns and candidate classes construct a segmentation-recognition candidate lattice, where each arc denotes a SP and each node denotes a character class assigned to a candidate pattern. Figure 6 shows a part of the candidate lattice of an example as shown in Fig. 4 where the correct path is thickly marked.

3.4 Handwritten Text Recognition by Optimal Path Search

The original path evaluation model was proposed in [6], and then formulated in [14]. We use the same criterion to evaluate the paths and search for the optimal text result by the Viterbi algorithm.

Given a handwritten text pattern, which is segmented into a sequence of candidate character patterns $X = x_1 x_2, \ldots, x_n$, and a candidate character pattern x_i is recognized as a character class c_i, the probability of forming a recognized text string $C = c_1 c_2, \ldots, c_n$ is calculated by the following evaluation function:

$$
\begin{aligned}
& f(X, C) \\
& = \sum_{i=1}^{n} \left(\begin{array}{l}
(\lambda_{11} + \lambda_{12}(k_i - 1)) \log P(c_i | c_{i-2}, c_{i-1}) + \\
(\lambda_{21} + \lambda_{22}(k_i - 1)) \log P(b_i | c_i) + \\
(\lambda_{31} + \lambda_{32}(k_i - 1)) \log P(q_i | c_i) + \\
(\lambda_{41} + \lambda_{42}(k_i - 1)) \log P(x_i | c_i) + \\
(\lambda_{51} + \lambda_{52}(k_i - 1)) \log P(\mathrm{p}_i^u | c_i) + \\
(\lambda_{61} + \lambda_{62}(k_i - 1)) \log P(\mathrm{p}_i^b | c_{i-1}, c_i) + \\
(\lambda_{71} \log P(g_{j_i} | Sb) + \\
\lambda_{72} \sum_{j=j_i+1}^{j_i+k_i-1} \log P(g_j | Sw)
\end{array} \right) + \lambda n
\end{aligned}
\tag{5}
$$

where n is the number of candidate character patterns in a path, k_i is the number of primitive segments within a candidate character pattern x_i, λ_{h1}, λ_{h2} (h=1∼7) and λ are weighting parameters, which are trained by MCE [15], to optimize the text recognition performance on a training dataset.

$P(c_i|c_{i-2}, c_{i-1})$ in Eq. (5) is tri-gram linguistic context probability. It smoothly combines the uni-gram, bi-gram and tri-gram by parameters, as shown in Eq. (6).

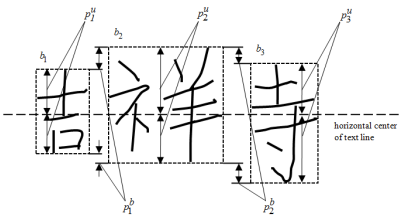
$$\begin{aligned} P(c_i|c_{i-2}, c_{i-1}) &= \beta_1 P(c_i|c_{i-2}, c_{i-1}) + \beta_2 P(c_i|c_{i-1}) \\ &\quad + \beta_3 P(c_i). \\ \text{subject to} \quad & \beta_1 + \beta_2 + \beta_3 = 1. \end{aligned} \tag{6}$$

The term b_i in Eq. (5) is a shape feature vector as shown in Fig. 7(a), which is composed of the height and width of the bounding box of a candidate character pattern.

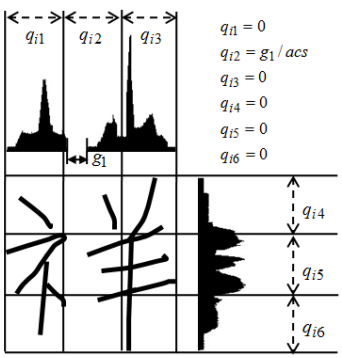
The term q_i in Eq. (5) is an inner-gap feature vector in a candidate character pattern, which is obtained by projecting the candidate character pattern into the vertical and horizontal directions, splitting each of their histograms into 3 slices, finding a gap or gaps in each slice, and summing total lengths of gaps as shown in Fig. 7(b).

The term p_i^u in Eq. (5) is an unary position feature vector, which consists of two vertical distances from the horizontal center of a text line to the top and bottom of the bounding box of a candidate character pattern, as shown in Fig. 7(a).

The term p_i^b in Eq. (5) is a binary position feature vector, which is composed of a vertical distance between the top edges of the bounding boxes of two adjacent candidate character patterns in a text line and that between the bottom edges of the bounding boxes, as shown in Fig. 7(a).



(a) b_i, p_i^u and p_i^b feature vectors.



(b) Inner-gap feature q_i.

Fig. 7 Geometric features of candidate character patterns.

We normalize the above 4 features (b_i, q_i, p_i^u, and p_i^b) by the acs (average character size).

We assume $P(b_i|c_i)$, $P(q_i|c_i)$, $P(p_i^u|c_i)$, and $P(p_i^b|c_{i-1}, c_i)$ to be normal distributions and model their logarithms by a quadratic discriminant function (QDF). They are together called geometric context.

$P(x_i|c_i)$ in Eq. (5) is given by a character recognizer detailed in Sect. 4.

$P(g_{j_i}|Sb)$ is the probability that the spacing between candidate character patterns (Sb) appears as g_{j_i} and $P(g_j|Sw)$ is the probability that spacing within a candidate character pattern (Sw) appears as g_j in Eq. (5).

The term g_j is a spacing feature vector concerning SP, which is extracted from an off-stroke. Both of the probabilities are approximated by the SVM model introduced in Sect. 3.2. Here, SP is always treated as Sb and NSP is always treated as Sw. UP is interpreted as either Sb or Sw. When it is within a character pattern, it is treated as Sw, when it is between character patterns, it is treated as Sb.

4. Experiments

We evaluate the proposed character-position-free handwritten Japanese text recognition method on the generated sample datasets, as well as the collected handwritten text patterns.

4.1 Datasets

We generate character-position-free handwritten text samples using the on-line handwritten Japanese text patterns from Kondate [16], which is a collection of handwritten text patterns from 100 people. First, we extract horizontally written text lines, and call it "Kondate_h". From Kondate_h, we delete text lines composed of more than 30 characters since long text would not be written in the car environment. Moreover, we divide a text line at the punctuation mark, and delete all punctuation marks. Then, we randomly change character positions in text patterns as described in Sect. 2, and generate Dataset M(M=1 to 4) of character-position-free handwritten text patterns.

On the other hand, we also collected text patterns written without supported by wrist or elbow and without visual feedback from 10 students as described in Sect. 2. We call this set of collected sample patterns as Dataset 5. Table 4 shows summaries of the datasets used for experiments.

For Datasets 1 to 4, we use a 4-fold cross-validation method to evaluate the performance of recognizers. For Dataset 5, however, we employ a simple method since it is too small to make a cross validation.

Table 4 Summaries of datasets.

Data	Text lines	Characters	Classes
Kondate_h	13,685	139,779	1,161
Dataset M (M=1 to 4)	15,389	129,076	1,123
Dataset 5	470	3,580	198

We divide each Dataset M (M=1 to 4) into 4 groups: Dataset_M_$D1$ (patterns by writer 1 ∼ writer 25), Dataset_M_$D2$ (writer 26 ∼ writer 50), Dataset_M_$D3$ (writer 51 ∼ writer 75), and Dataset_M_$D4$ (writer 76 ∼ writer 100), each group includes 25 peoples' patterns, and let 3 groups (75 peoples' patterns) as training patterns, the remaining 1 group (25 peoples' patterns) as testing patterns. Moreover, to let our proposed model recognize patterns of all the datasets with the same set of parameters, we combine all the training patterns (75×4) as the total training patterns. That is, the combined training patterns are shared among the following experiments but the testing patterns are used from each Dataset. If we use the training patterns and testing patterns in each dataset, we get slightly better results but it seems unfair since we cannot predict which model is appropriate.

4.2 Results of Experiments

We compare CSM and USM with the original recognizer. It was developed for normally handwritten horizontal Japanese text patterns based on the method [6] and reduced in size by feature selection, LDA, vector quantization and data type transformation. This comparison is made by changing over-segmentation and replacing the candidate character segmentation probability while succeeding the character recognition, unary and binary position features, size and inner-gap features, and linguistic context to evaluate the candidate segmentation-recognition paths. In addition, parameters in the path evaluation function are retrained on the combined training patterns by MCE.

To evaluate the effectiveness of the segmentation probability by the SVM model, we consider the cases when the segmentation scores are not combined in the path evaluation function for Viterbi search, namely, by setting 0 for λ_{71} and λ_{72} in Eq. (5), for CSM and USM. Then we adjust the other parameters in Eq. (5) on the combined training patterns by MCE. We call them the candidate segmentation method without segmentation scores (CSM_w/o Ss) and the undecided segmentation method without segmentation scores (USM_w/o Ss), respectively.

As for the character recognizer, which combines online and off-line character recognizers by a linear function [17], where the combing parameters are trained by Nakayosi [18], we keep top 10 candidate character classes for every candidate character pattern. We also use Nakayosi to train geometric feature functions: unary and binary position features, size and inner-gap features.

As for the linguistic context model, it is trained on the year 1993 volume of the ASAHI newspaper and the year 2002 volume of the NIKKEI newspaper. We estimate the smoothing parameters ($\beta_1 = 0.7, \beta_2 = 0.2, \beta_3 = 0.1$) in Eq. (6) by Nakayosi.

As for the SVM model, we train it on off-strokes of the combined training patterns. However, the number of off-strokes is so large (more than a million), we use 1/10 of them as training data.

To evaluate the text recognizers, we use the recognition rate (R_c) defined in Eq. (7), the segmentation measure (F) defined in Eq. (8), which combines recall and precision rates, and the average recognition time cost per character pattern ($T_{av\_rec\_c}$).

All the experiments are made on a PC with Intel(R) Core^{TM} i7-3770 CPU @3.40GHz 3.40GHz (2 processers) and 8 GB memory.

$$R_c = \frac{number\ of\ correctly\ recognized\ characters}{number\ of\ all\ characters} \tag{7}$$

$$\begin{aligned} F &= \frac{2}{1/R + 1/P} \\ R &= \frac{\text{number of correctly detected SPs}}{\text{number of true SPs}} \\ P &= \frac{\text{number of correctly detected SPs}}{\text{number of detected SPs (including false)}} \end{aligned} \tag{8}$$

Table 5 to Table 8 shows the recognition performance for Dataset 1 to Dataset 4, respectively.

For Dataset 5 (collected sample patterns), we firstly prepare the combined text patterns: {Dataset_M_$D1$,

Table 5 Recognition performance on Dataset 1.

Method \ Performance	R_c	F	$T_{av\_rec\_c}$
Original recognizer	23.72%	0.4902	0.012 s
CSM	89.95%	0.9671	0.089 s
USM	91.83%	0.9751	0.139 s
CSM_w/o Ss	89.72%	0.9646	0.086 s
USM_w/o Ss	91.37%	0.9693	0.125 s

Table 6 Recognition performance on Dataset 2.

Method \ Performance	R_c	F	$T_{av\_rec\_c}$
Original recognizer	39.61%	0.6627	0.013 s
CSM	90.99%	0.9728	0.078 s
USM	92.23%	0.9771	0.121 s
CSM_w/o Ss	90.81%	0.9707	0.075 s
USM_w/o Ss	91.80%	0.9722	0.100 s

Table 7 Recognition performance on Dataset 3.

Method \ Performance	R_c	F	$T_{av\_rec\_c}$
Original recognizer	0.00	0.00	N/A
CSM	91.31%	0.9784	0.299 s
USM	92.34%	0.9795	0.585 s
CSM_w/o Ss	91.09%	0.9760	0.283 s
USM_w/o Ss	91.85%	0.9733	0.557 s

Table 8 Recognition performance on Dataset 4.

Method \ Performance	R_c	F	$T_{av\_rec\_c}$
Original recognizer	0.04%	0.1600	N/A
CSM	83.26%	0.9273	0.111 s
USM	84.97%	0.9325	0.133 s
CSM_w/o Ss	82.94%	0.9274	0.111 s
USM_w/o Ss	84.01%	0.9294	0.128 s

Dataset_M_$D2$, Dataset_M_$D3$} (M=1 to 4), to train both the segmentation methods and to tune the parameters of the path evaluation function by MCE; then, we increase the number of candidate character classes for each candidate character pattern from 10 to 15 (top 15), further update the parameters of the path evaluation function with adding 3/5 of collected text patterns by the genetic algorithm [6], and test the performance to the remaining 2/5. Table 9 shows the recognition performance for Dataset 5.

The final experiment is to confirm the performance of CSM and USM on normally handwritten text patterns. Table 10 shows the results. For comparison, we also show the performance of the original recognizer (which is CSM) and that with CSM being replaced by USM, i.e., all off-strokes being set as UP (named Original recognizer with USM). Note that parameters for these two recognizers have been tuned for normally handwritten text.

Table 10 shows interesting results. When the parameters are tuned for normally handwritten text, CSM (Original recognizer) is significantly more accurate and quicker than USM (Original recognizer with USM). On the other hand, when the parameters are tuned for character-position-free handwritten text, USM is more accurate with slightly higher time cost. This will be discussed in the next subsection.

4.3 Analysis and Discussion

From the results presented in the previous Sect. 4.2, both CSM and USM produce far better recognition rate and segmentation measure for all the datasets than the original recognizer.

Since the average recognition rate by the original recognizer on normally handwritten text patterns is 93.40%, both CSM and USM have realized almost the similar recognition rate even for character-position-free handwritten text except Dataset 4.

In fact, it is notable that both CSM and USM achieve 90.38% and 92.04% recognition rate for Kondate_h, respectively, as shown in Table 10. This implies that many factors in the evaluation function (Eq. (5)) may relax one or two constraints, especially, the segmentation constraint at the sacrifice of the time complexity.

Table 9 Recognition performance on 2/5 of Dataset 5.

Method \ Performance	R_c	F	$T_{av\_rec\_c}$
Original recognizer	50.84%	0.7670	0.015 s
CSM	74.86%	0.9033	0.119 s
USM	76.19%	0.9046	0.159 s

Table 10 Recognition performance on Kondate_h.

Method \ Performance	R_c	F	$T_{av\_rec\_c}$
Original recognizer	93.40%	0.9917	0.012 s
Original recognizer with USM	90.04%	0.9542	0.050 s
CSM	90.38%	0.9711	0.050 s
USM	92.04%	0.9814	0.070 s

The original recognizer [6] is the integrated segmentation and recognition system incorporating several factors into the evaluation function. It misclassifies true SPs as NSPs in the over-segmentation step for the character-position-free on-line handwritten text recognition, however, due to the fully or partially overlaid characters. While USM sets each off-stroke as UP to avoid the misclassification and CSM keeps the true SPs as much as possible by adjusting the thresholds for the output of SVM. That is why these two methods can overcome the problem of the original recognizer for the character-position-free on-line handwritten text recognition.

As for the effectiveness of the segmentation probability by the SVM model, it is clarified that unemployment of the segmentation scores decreases the recognition rate of USM by 0.46 point, 0.43 point, 0.49 point and 0.96 point for Dataset 1 to Dataset 4, respectively, and that of CSM by 0.23 point, 0.18 point, 0.22 point and 0.32 point for Dataset 1 to Dataset 4, respectively. The recognition rate of CSM decreases less than USM, because the former classifies off-strokes into NSP, SP, and UP based on the output of the SVM model. Hence, it is better to combine the segmentation scores in the path evaluation for Viterbi search.

As for the recognition speed, except Dataset 3, the average recognition time cost per character pattern is 0.093 second and 0.131 second, by CSM and USM, respectively, whereas the original recognizer takes 0.012 second on normally handwritten text. This is because most off-strokes are classified into UPs in CSM, and all off-strokes are set to UPs, so that the constructed candidate lattice becomes so large.

Moving to the comparison between CSM and USM, the recognition rate by the latter is superior to the former by 1.88 point, 1.24 point, 1.03 point, and 1.71 point, for Dataset 1 to Dataset 4, respectively. On the other hand, the average time cost per character by the latter increases 0.050 second, 0.043 second, 0.286 second, and 0.022 second, for Dataset 1 to Dataset 4, respectively, which is about 1.20 ~ 1.96 times the former, but it is not a problem for real-time recognition.

Dataset 3 is a set of text patterns where characters are completely overlaid. The overlaid handwritten character recognition has been proposed for small surface devices. Our experiments show that it can be treated uniformly by the character-position-free handwritten text recognition.

The recognition rate is even slightly higher for Dataset 3. The off-strokes between characters move generally from bottom-right to top-left, so that character segmentation reliability could be enhanced and the character recognition rate is improved slightly from other cases.

On the other hand, the time cost is about 4 times larger than the others. The reason is as follows: candidate character patterns whose widths are longer than the threshold are deleted from the candidate lattice as described in Sect. 3.1, but no candidate is deleted for Dataset 3. Even in this case, however, the time is about 0.3 or 0.6 second for a character. This is not a serious problem for practical applications since

it is far quicker than handwriting by people.

For Dataset 5 of collected handwritten text samples, the recognition rate is improved from the original recognizer but still lower than the others. This would be partially because the dataset is too small and partially because some strokes in character patterns are so largely displaced or deformed so that those patterns are not character-position-free but stroke-position-free. The amount of those patterns is not large but not negligible. Removing all the position information could be considered but would damage total character recognition considerably. To cope with them is our next challenge.

As for the comparison between USM and CSM in general, USM is more accurate with slightly higher time cost for character-position-free handwritten text, but USM is inferior to CSM for normally handwritten text patterns in both accuracy and speed as shown in Table 10. This was exactly the reason that we employed CSM for ordinary text input. When the character position information is reliable, it must be exploited.

5. Conclusion

In this paper, we have proposed a method to recognize character-position-free on-line handwritten text patterns. We have considered two segmentation methods, one classifies each off-stroke into NSP, SP or UP according to the output of SVM, the other sets each off-stroke as UP. The results of experiments confirmed that the proposal achieves the best recognition performance for character-position-free text patterns by the undecided segmentation method (USM), approaching the performance of the original recognizer on normally handwritten text patterns. The method has been employed for the EMIRAI 3 xDAS assisted-driving concept car.

We have also considered the case that characters are completely overlaid in this framework and have shown that the proposed method works well.

Some work remains to be done. Firstly, we need to collect a large set of real patterns in automobile environment. Although we simulated physical conditions to write characters, mental conditions while driving a car with watching the frontal view and predicting the other objects' movement could not be reflected to simulated patterns. Real patterns must be also collected from small-surface environment. Secondly, if we could remove non-character patterns from the candidate lattice reliably, we could improve the recognition speed. Thirdly, we must enhance the method to recognize stroke-position-free character patterns by extracting more geometric and linguistic features.

Acknowledgments

This research is partially being supported by Grant-in-aid for Scientific Research C-15K00225.

References

[1] M. Nakagawa, J. Tokuno, B. Zhu, M. Onuma, H. Oda, and A. Kitadai, "Recent results of online Japanese handwriting recognition and its applications," Arabic and Chinese Handwriting Recognition (D. Doermann and S. Jaeger ed. selected papers from SACH 2006), Lect. Notes Comput. Sci. (LNCS) 4768, pp.170–195, 2008.

[2] C.C. Tappert, C.Y. Suen and T. Wakahara, "The state of the art in on-line handwriting recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol.12, no.8, pp.787–808, Aug. 1990.

[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] B. Zhu and M. Nakagawa, "Recent trends in online handwritten character recognition," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J95-D, no.4, pp.335–340, April 2012.

[5] H. Murase, T. Wakahara, and M. Umeda, "Online writing-box free character string recognition by candidate character lattice method," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J68-D, no.4, pp.765–772, April 1985.

[6] B. Zhu, X.-D. Zhou, C.-L. Liu, and M. Nakagawa, "A robust model for on-line handwritten Japanese text recognition," Int. J. Document Analysis and Recognition, vol.13, no.2, pp.121–131, June 2010.

[7] H. Shimodaira, T. Sudo, M. Nakai, and S. Sagayama, "On-line overlaid handwriting recognition based on substroke HMMs," Proc. 7th ICDAR, pp.1043–1047, Edinburgh, Scotland, Aug. 2003.

[8] Y. Tonouchi and A. Kawamura, "Text input system using online overlapped handwriting recognition for mobile devices," Proc. 9th ICDAR, pp.754–758, Curitiba, Brazil, Sept. 2007.

[9] A. Bharath and S. Madhvanath, "FreePad: a novel handwriting-based text input for pen and touch interfaces," Proc. 13th Int. Conf. Intelligent User Interfaces, pp.297–300, Canary Island, Spain, Jan. 2008.

[10] Y. Zou, Y. Liu, Y. Liu, and K. Wang, "Overlapped handwriting input on mobile phones," Proc. 11th ICDAR, pp.369–373, Beijing, China, Sept. 2011.

[11] X. Wan, C. Liu, and Y. Zou, "On-line Chinese character recognition system for overlapping samples," Proc. 11th ICDAR, pp.799–803, Beijing, China, Sept. 2011.

[12] Y.-F. Lv, L.-L. Huang, D.-H. Wang, and C.-L. Liu, "Learning-based candidate segmentation scoring for real-time recognition of online overlaid Chinese handwriting," Proc. 12th ICDAR, pp.74–78, Washington, DC, Aug. 2013.

[13] B. Zhu and M. Nakagawa, "Segmentation of on-line freely written Japanese text using SVM for improving text recognition," IEICE Trans. Inf. & Syst., vol.E91-D, no.1, pp.105–113, Jan. 2008.

[14] J. Gao, B. Zhu, and M. Nakagawa, "Development of a robust and compact on-line handwritten Japanese text recognizer for hand-held devices," IEICE Trans. Inf. & Syst., vol.E96-D, no.4, pp.927–938, April 2013.

[15] B.-H. Juang, W. Chou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition," IEEE Trans. Speech and Audio Processing, vol.5, no.3, pp.257–265, May 1997.

[16] T. Matsushita and M. Nakagawa, "A database of On-line handwritten mixed objects named "Kondate"," Proc. 14th ICFHR, pp.369–374, Crete, Greece, Sept. 2014.

[17] B. Zhu, J. Gao, and M. Nakagawa, "Objective function design for MCE-based combination of on-line and off-line character recognizers for on-line handwritten Japanese text recognition," Proc. 11th ICDAR, pp.594–599, Beijing, China, Sept. 2011.

[18] M. Nakagawa and K. Matsumoto, "Collection of on-line handwritten Japanese character pattern databases and their analysis," International J. Document Analysis and Recognition, vol.7, no.1, pp.69–81, Sept. 2004.

[19] JAMA (Japan Automobile Manufacturers Association, Inc.), "Guidelines for in-vehicle display systems-version 3.0,", Aug. 2004.

[20] Mitsubishi electric, "Mitsubishi electric introduces EMIRAI 3 xDAS assisted-driving concept car," 44th Tokyo Motor Show, Tokyo big sight, Japan, 2015.
http://www.mitsubishielectric.com/news/2015/pdf/1008.pdf

Jianjuan Liang received the B.Sc. and M.Sc. degrees in Computer Science and Technology from Guizhou University, Guiyang, China, in 2007 and 2010, respectively. She is currently pursuing a Ph.D. degree in Engineering from Tokyo University of Agriculture and Technology (TUAT), Japan. Her research interests include handwritten character recognition and text recognition.

Bilan Zhu was born on 13 December 1974 in China. She received B. Sc. and M.Sc. degrees from Tokyo University of Agriculture and Technology (TUAT) in 2003 and 2004, respectively. In 2007, she received Ph.D. degree in Engineering from TUAT. From April 2007, she has been working at Tokyo University of Agriculture and Technology. Currently, she is an Assistant Professor of Media Interaction in Department of Computer and Information Sciences. She is working on on-line, off-line handwriting character recognition, documents processing and context analysis of Japanese, Chinese and English text.

Taro Kumagai received M.Sc. degree from Kyushu University, Japan in 2006. In 2009, he received Ph.D. degree from Kyushu University. From 2009, he has been with Mitsubishi Electric Corp., Japan. He is working on in-vehicle human machine interface system.

Masaki Nakagawa was born on 31 October 1954 in Japan. He received B. Sc. and M. Sc. degrees from the University of Tokyo in 1977 and 1979, respectively. During the academic year 1977/78, he followed Computer Science course at Essex University in England, and received M.Sc. with distinction in Computer Studies in July 1979. He received Ph.D. degree in Information Science from the University of Tokyo in December 1988. From April 1979, he has been working at Tokyo University of Agriculture and Technology. Currently, he is a Professor of Media Interaction in Department of Computer and Information Sciences.