A Robust Method for Coarse Classifier Construction from a Large Number of Basic Recognizers for On-line Handwritten Chinese/Japanese Character Recognition

Bilan Zhu and Masaki Nakagawa

Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology

2-24-16 Naka-cho Koganei-shi, Tokyo, 184-0012, Japan

E-mail: {[zhubilan, nakagawa](mailto:zhubilan@cc.tuat.ac.jp)}@cc.tuat.ac.jp

Corresponding author: Bilan Zhu

Tel: 0081-423-88-7490 Fax: 0081-423-88-7490

zhubilan@cc.tuat.ac.jp

Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology

2-24-16 Naka-cho Koganei-shi, Tokyo, 184-8588, Japan

Abstract: In this paper, a systematic method is described that constructs an efficient and robust coarse classifier from a large number of basic recognizers obtained by different parameters of feature extraction, different discriminant methods or functions, etc. The architecture of the coarse classification is a sequential cascade of basic recognizers that reduces the candidates after each basic recognizer. A genetic algorithm determines the best cascade with the best speed and highest performance. The method was applied for on-line handwritten Chinese and Japanese characters recognitions. We produced hundreds of basic recognizers with different classification costs and different classification accuracies by changing parameters of feature extraction and discriminant functions. From these basic recognizers, we obtained a rather simple two-stage cascade, resulting in the whole recognition time being reduced largely while maintaining classification and recognition rates.

Keyword *On-line character recognition; Chinese character recognition; Japanese character recognition; Coarse classifier; Genetic algorithm.*

1. Introduction

Due to the development and proliferation of pen-based or touch-based input devices such as tablet terminals, smart phones, electronic whiteboards, and digital pens (e.g., the Anoto pen), on-line handwritten character recognition is receiving even more attention than before.

Although character classifiers with high recognition accuracy have been reported [1-5], the demand for speeding up recognition is very high for portable devices as well as for desktop applications for which handwriting recognition is incorporated as one of the modules. The performance of these relatively small devices requires having a fast as possible recognition speed while maintaining high accuracy. Even for a desktop PC with relatively high performance, a recognition speed of within 0.5 seconds per page is required in actual applications. Therefore, we need to refine the recognition scheme to improve the processing speed.

Chinese, Japanese, or Korean have thousands of different categories, and their large character set is problematic not only for the recognition rate but also for the recognition speed. A general approach to improving the recognition speed is to perform coarse classification, pre-classification, or candidate selection before the fine classification [6, 7].

The coarse classification typically uses simpler classification algorithms or fewer features in order to achieve a better speed than does the fine classification. It is used to select a relatively small subset of candidates out of a very large set quickly. The fine classification would then be used on these candidates to match an input pattern so that the whole recognition time is reduced. Current approaches for coarse classification typically use distance measures that are simpler than those for fine classification [8, 9]. The confidence evaluation provides even more precise candidate selection [10]. Others use simple features different from those for fine classification [11].

Sequential (multi-stage) classifications using a partial set of features at each stage have also been used [12]. A common characteristic of these methods is that the entire candidate selection process is performed during the recognition process. We call these “dynamic approaches.”

In contrast to dynamic approaches, prototypes may be organized prior to the search itself so that the search is performed on a subset of prototypes. We could mention a number of methodologies that vary slightly in how the data is organized. The simplest ones are proposals for ordered spaces and tree structures. The search on pre-structured spaces aims particularly at alleviating problems with search costs. As a result, recognition is accelerated. We refer to these methods as “static approaches.” In previous research within our group, we applied a static approach named “structuring

search space” (SSS) for an off-line recognizer of handwritten Japanese characters [13, 14]. In SSS, prototypes are organized by unsupervised clusters, and their centroids, for example, are used to represent the group. During recognition, an input pattern is first compared with all of the clusters’ centroids, and second, only the clusters with centroids similar to the input are used as search space. This method was extended to a two-layered search space [15] and named the “layered search spaces (LSS) method,” which will be detailed later. Being different from the approach by Tseng et al. [16] and that by Fujimoto et al. [17], the LSS method works in the original feature space for fine classification and, therefore, only has to assume one distance space. Waizumi et al. also presented a multilayer search space construction method for character class prototypes that uses LVQ [18].

In general, the previous methods for both the dynamic and static approaches have applied rather intuitive ideas for selecting simple recognition functions such as Euclidean distance or fewer features to speed up recognition. Therefore, they have to take a larger number of output candidates to maintain recognition rates, which causes recognition time to increase and limits the effects on speedup.

In this paper, we present a systematic method for constructing an efficient and robust coarse classifier that uses a genetic algorithm (GA) for the on-line recognition of handwritten Chinese and Japanese characters. By preparing a sufficient number of basic recognizers, placing them in cascade architecture and employing GA to find the best components and parameters, we can find a better coarse classifier than previously employed methods. We can create many basic recognizers with different classification costs and different classification accuracies by controlling the parameters of feature extraction and discriminant functions which have been used for fine classification such as the Modified quadratic discriminant function (MQDF). If we set the parameters different from their optimal values for fine classification, it may degrade the top recognition rate but speed up recognition with smaller memory cost, and retain the correct candidate within top- N by setting N large enough. We previously tried to find a combination of several base classifiers heuristically, but we were quite unsure on whether we had obtained the best combination. To solve the problem, we have moved to propose a method to construct a coarse classifier based on a cascade of many basic recognizers.

For on-line handwritten Chinese and Japanese characters, we prepare 243 basic recognizers with different classification costs and different classification accuracies by controlling the parameters of feature extraction and discriminant functions as well as the LSS method. Then, we construct a sequential cascade of basic recognizers and reduce the candidates after each basic recognizer. The parameters for the cascade of basic recognizers are estimated by using GA so as to optimize the holistic character recognition performance. The more the basic recognizers are, the better the accuracy of the cascade

of basic recognizers can be optimized by GA which minimizes runtime while maintaining the recognition accuracy. The resultant architecture may be made of a few components, but we are more confident that it is best among many choices. The coarse classifier thus developed follows the dynamic approach although it integrates the statistically tuned LSS. Experimental results for the CASIA-OLHWDB1.1 database (on-line Chinese handwriting database) [19] and those for the TUAT Nakayosi/Kuchibue database (on-line Japanese handwriting database) [20] demonstrate the superiority of our method.

This paper is an extension to the conference paper [21]. We added more details of background and objective, extended technical description and also added extensive experimental results by applying our coarse classification method to three fine recognizers for the above Chinese and Japanese handwriting databases to demonstrate the applicability of our proposed method. The rest of this paper is organized as follows. In Section 2, an overview of our on-line handwritten character recognition system is presented. In Section 3, a linear structure for constructing a coarse classifier is designed, and in Section 4, a parameter optimization method is described. In Section 5, experimental results are presented, and in Section 6, our conclusion is presented.

2. Recognition System Overview

2.1 Recognition System

We process each on-line character pattern as shown in Fig. 1. There are thousands of categories in the Chinese and also in the Japanese language. First, to improve the recognition speed, we reduce recognition candidates by using a coarse classifier for each on-line character input pattern. Then, we select a smaller category set from the candidates' output by using a fine classifier.

Figure 1. On-line handwritten character recognition. First, recognition candidates are reduced by using a coarse classifier. Then, a smaller category set is selected by using a fine classifier.

The fine classifier is in fact one of the modules used for handwritten text recognition [1], and it is combined with a segmentation module, geometric context processor, and a linguistic postprocessor.

2.2 Fine Classification

Fine classification after coarse classification can be a structural recognizer [4, 5] that uses structural features such as sampling points, line segments and/or strokes, or an un-structural recognizer [3] that uses un-structural features such as directional features and gradient histogram features. For on-line recognition, structural features are often employed with hidden Markov models (HMMs) or Markov random field (MRF) [4, 5]. Since un-structural features are easily extracted from an on-line handwritten pattern by discarding temporal and structural information, we can also apply an un-structural recognizer [3]. The modified quadratic discriminant function (MQDF) [22] is often employed for Japanese and Chinese character recognition. Structural recognizers and un-structural recognizers have their advantages and disadvantages. For justifying our proposed approach for coarse classification and demonstrating the applicability of the idea, we apply our coarse classification method to three fine recognizers that are a structural MRF recognizer, a structural HMM recognizer and an un-structural MQDF recognizer for both the CASIA-OLHWDB1.1 database and the TUAT Nakayosi/Kuchibue database.

For the structural recognizers, we extract feature points along the pen-tip trace from pen-down to pen-up. We use the coordinates of feature points as unary features and the differences in coordinates between the neighbouring feature points as binary features. We then use a MRF model or a HMM model [5] to match the feature points with the states of each character class of the input candidates produced by the coarse classifier and obtain similarity for each character class. We then select the top character categories with the largest similarities as the output candidates of the fine classifier. For the HMM recognizer, we merged the binary features into the unary features and used a vector of larger dimension for each single-site because HMMs do not use the binary features for each pair-site and only use the unary features for each single-site [5].

For the un-structural MQDF recognizers, we do not need to transform each on-line character pattern to an off-line character pattern (two dimensional images), but extract directional features: histograms of normalized stroke direction [3] directly from the on-line character pattern. For coordinate normalization, we apply pseudo 2D bi-moment normalization (P2DBMN) [23]. The local stroke direction is decomposed into eight directions, and from the feature map of each direction, 8x8 values are extracted by Gaussian blurring so that the dimensionality of feature vectors is 512. To improve the Gaussianity of feature distribution, each value of the 512 features is transformed by the Box-Cox transformation (also

called variable transformation). The input feature vector is reduced from 512D to n D by the Fisher linear discriminant analysis (FLDA). Then we use the n D feature vectors to create a MQDF recognizer as follows:

$$g_2(\mathbf{x}, \omega_i) = \sum_{j=1}^k \frac{1}{\lambda_{ij}} [\boldsymbol{\varphi}_{ij}^T (\mathbf{x} - \boldsymbol{\mu}_i)]^2 + \frac{1}{\delta} \{ \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 - \sum_{j=1}^k [\boldsymbol{\varphi}_{ij}^T (\mathbf{x} - \boldsymbol{\mu}_i)]^2 \} + \sum_{j=1}^k \log \lambda_{ij} + (n - k) \log \delta \quad (1)$$

where μ_i is the mean vector of class ω_i , λ_{ij} ($j = 1, \dots, k$) are the largest eigenvalues of the covariance matrix and φ_{ij} are the corresponding eigenvectors, k denotes the number of principal axes, and δ is a modified eigenvector that is set as a constant. The value of δ can be optimized on the training data set. However, for a convenience, we simply set it as $\gamma \lambda_{average}$ where $\lambda_{average}$ is the average of λ_{ij} ($i, j = 1, \dots, n$) for all features of all classes and γ is a constant that is larger than 0 and smaller than 1.

According to previous works [3, 24], the best un-structural recognition performances are obtained when n is about 160 and k is about 50 in the MQDF recognizers for both the two databases (CASIA-OLHWDB1.1, TUAT Nakayosi/Kuchibue). Therefore, for the un-structural MQDF recognizers for the two databases, we take n as 160 and k as 50, respectively.

3. Linear Structure Design for Constructing a Coarse Classifier

In this section, we first describe preprocessing and feature extraction. We then describe basic character recognizers, which form a coarse classifier. Finally, we present a linear structure design for coarse classification that reduces candidates one after another by using a cascade of basic recognizers.

3.1 Preprocessing and Feature Extraction

For the coarse classifier, we extract the same 512 features from each on-line character pattern as the MQDF fine recognizer described in Section 3.2.

The coarse classifier in the on-line recognizer can use not only un-structural features but also structural features. Therefore, we also extract five features from each on-line character pattern, which have been effectively applied to our coarse classification step. After the P2DBMN normalization, we extract feature points by using the method by Ramner

[25]. First, the start and end points of every stroke are picked up as feature points. Then, the most distant point from the straight line between the adjacent feature points is selected as a feature point if the distance to the straight line is greater than a threshold value. This selection is done recursively until no more feature points are selected. The feature point extracting process is shown in Fig. 2. After we extract feature points, we extract five features: the number of strokes, X-direction stroke length, Y-direction stroke length, the number of X-direction change times, and the number of Y-direction change times, as shown in Fig. 3. We finally obtain 517 features from an on-line character pattern. Then, to improve the Gaussianity of feature distribution, each value of the 517 features is transformed by Box-Cox transformation.

Figure 2. Feature points extraction. First, the start and end points of every stroke are picked up as feature points. Then, the most distant point from the straight line is selected as a feature point. This selection is done recursively.

Figure 3. On-line character features. the number of strokes, X-direction stroke length, Y-direction stroke length, the number of X-direction change times, and the number of Y-direction change times.

3.2 Basic Character Recognizers

The input feature vector is first reduced from 517D to n D by FLDA. Then, we can use the n D feature vectors to create a MQDF recognizer [23] as shown in (1).

We investigated the results of different values for the MQDF parameter k and feature extraction FLDA parameter n in fine recognitions by MQDFs using the on-line Chinese handwriting database (CASIA-OLHWDB1.1 and the on-line Japanese handwriting database (TUAT Nakayosi/Kuchibue). In the CASIA-OLHWDB1.1 database, we employed character patterns from 240 writers (898,516 Chinese character patterns of 3,755 classes) for training and those from 60 writers (196,439 Chinese character patterns of 3,755 classes) for testing. In the CASIA-OLHWDB1.1 database, each writer and each character class have almost the same numbers of sample patterns. For Japanese, we used the TUAT Nakayosi database for training and the TUAT Kuchibue database for testing. Table 1 shows the details of the databases. Each character class has a different number of sample patterns, and the kana and symbol classes have more patterns (see Table 1). To maintain balance, we selected 120 patterns at random from each character class of the Kuchibue database and used the same number of sample patterns for each character class to evaluate the performance. The experiments were implemented on an Intel(R) Xeon(R) CPU W5590 @ 3.36 GHz 3.36 GHz (2 processors) with 12 GB memory.

Table 2 shows the results, where the numbers without any parentheses or brackets are the recognition rates, those shown in parentheses are the average character recognition time and those shown in brackets are the memory costs. For an input pattern, for accelerating recognitions, we first select 100 top rank candidate classes by a coarse classifier using the Euclidean distance to class means.

Table 1. Statistics of character pattern databases

		Kuchibue_d	Nakayosi_t
#writers		120	163
#characters /each writer	Total	11,962	10,403
	Kanji/Kana/Symbol/alpha numerals	5,643/5,068/1,085/166	5,799/3,723/816/65
#character categories /each writer	Total	3,356	4,438
	Kanji/Kana/Symbol/alpha numerals	2976/169/146/62	4058/169/149/62
#average category characters	Total	3.6	2.3
	Kanji/Kana/Symbol/alpha numerals	1.9/30.0/7.4/2.7	1.4/22.0/5.5/1.0

Table 2. Recognition rates (%), average character recognition time (ms) and memory cost (MB) for k and n

$n \backslash k$	20	40	60	80	100	120	140	160	180
70				89.8(1.3) [42]	91.3(1.6) [52]	91.7(2.0) [62]	91.9(2.3) [72]	91.9(2.6) [82]	91.9(2.8) [93]
60			87.5(0.9) [27]	90.9(1.2) [36]	91.8(1.5) [45]	92.1(1.7) [53]	92.2(2.1) [62]	92.2(2.3) [71]	92.2(2.5) [80]
50			89.3(0.9) [23]	91.4(1.0) [30]	92.1(1.3) [37]	92.4(1.5) [45]	92.4(1.8) [52]	92.4(1.9) [59]	92.4(2.2) [66]
40		86.0(0.5)[12]	90.1(0.7) [18]	90.5(0.6) [24]	92.2(1.1) [30]	92.3(1.3) [36]	92.5(1.5) [42]	92.5(1.6) [48]	92.5(1.9) [53]
30		87.1(0.4) [9]	90.5(0.6) [14]	90.6(0.5) [18]	92.2(0.9) [23]	92.3(1.1) [27]	92.4(1.2) [32]	92.5(1.4) [36]	92.5(1.5) [40]
20	75.4(0.2)[3]	87.6(0.3) [6]	90.6(0.5) [9]	90.0(0.4) [12]	91.4(0.5) [15]	92.2(0.8) [18]	92.3(0.9) [21]	92.3(1.1) [24]	92.4(1.2) [27]
10	74.8(0.2)[2]	87.4(0.3) [3]	90.0(0.4) [5]	85.9(0.3) [6]	87.8(0.4) [8]	91.7(0.6) [10]	91.9(0.7) [11]	92.0(0.8) [13]	92.0(0.9) [14]
70				93.6(1.4) [49]	95.2(1.7) [61]	94.7(2.0) [73]	94.9(2.2) [85]	95.0(2.5) [97]	95.0(2.8) [109]
60			92.6(0.9) [32]	93.9(1.2) [42]	94.8(1.5) [53]	94.8(1.7) [63]	95.0(2.0) [73]	95.1(2.3) [84]	95.1(2.5) [94]
50			93.6(0.8) [27]	94.6(1.1) [35]	95.0(1.3) [44]	95.0(1.5) [53]	95.1(1.8) [61]	95.2(2.0) [70]	95.2(2.2) [79]
40		92.7(0.5) [15]	94.6(0.7) [22]	94.8(0.9) [28]	95.0(1.1) [35]	95.0(1.3) [42]	95.1(1.5) [49]	95.1(1.7) [56]	95.1(1.9) [63]
30		92.9(0.5) [11]	94.7(0.6) [16]	94.9(0.8) [22]	95.0(0.9) [27]	95.1(1.1) [32]	95.1(1.3) [37]	95.1(1.4) [43]	95.1(1.6) [48]
20	87.6(0.2) [4]	93.6(0.4) [7]	94.8(0.5) [11]	94.7(0.6) [15]	94.9(0.8) [18]	95.0(0.9) [22]	95.0(1.0) [25]	95.1(1.1) [29]	95.1(1.3) [32]
10	84.5(0.2) [2]	93.5(0.3) [4]	94.7(0.4) [6]	94.6(0.5) [8]	94.8(0.6) [9]	94.8(0.7) [11]	94.8(0.8) [13]	94.9(0.9) [15]	94.9(0.9) [17]

Upper: CASIA-OLHWDB1.1; lower: TUAT Nakayosi/Kuchibue; numbers without any parentheses or brackets: recognition rates (%); numbers in parentheses: average character recognition time (ms); numbers in brackets: memory costs (MB).

From the results we can see the best performances are obtained when n is about 160 and k is about 30 for the CASIA-OLHWDB1.1 database and n is about 160 and k is about 50 for the TUAT Nakayosi/Kuchibue database. When n and k are smaller than their optimal values, even though the recognizer's top recognition rate is degraded, it can recognize an input pattern with higher speed.

Here, we introduce the cumulative rate as the rate by which the correct class is listed within the top- N candidates by the recognizer. For coarse classification, we set N to be large so that a sufficiently high cumulative rate is assured and the correct class is passed to the fine classification. For all k and n in Table 2, the cumulative rates with more than 99.0% were obtained within the top-100 candidates.

For MQDF recognizers as coarse classifiers with smaller n and k than their optimal values, we set a larger N automatically in order to retain the high cumulative rate, say 99.9 %.

Namely, if we set n and k smaller than their optimal values, it may degrade the top recognition rate but speed up recognition with a smaller memory cost and retain the correct candidate within the top- N by setting N to be large enough. Without experiments, we do not know which recognizer is the best for coarse classification.

The other group of recognizers are produced from the LSS method [13-15]. The basic concept is to use pivots in the search space of character pattern prototypes. Given an input feature vector, it is compared only with the pivots, and those close to it are selected. Then, it is matched with prototypes close to the selected pivots. We introduce two layers. An input feature vector is compared with the top-layer pivots, and those close to it are selected. Then, it is compared with the 2nd-top-layer pivots close to the selected top-layer pivots, and a set of candidates close to the selected 2nd-top-layer pivots are selected. For the LSS, we can also use different feature dimensionality n . Under the condition that the top- N cumulative rate is more than 99.9%, n smaller than the optimal value brings a lower processing cost and a larger output candidate set while the larger n , unless it is larger than the optimal value, brings higher recognition costs and a smaller output candidate set.

We can create many basic recognizers from the MQDF and LSS by adjusting the feature extraction FLDA parameter n and the MQDF parameter k .

3.3 Linear Structure Design

We create many basic recognizers of the MQDF and LSS by adjusting the FLDA parameter n and the MQDF parameter k as follows.

$$\begin{aligned} n[M_n] &= \{v^n_1, v^n_2, \dots, v^n_{M_n}\} \\ k[M_k] &= \{v^k_1, v^k_2, \dots, v^k_{M_k}\} \end{aligned}$$

where each value is denoted as $n[l]$ ($l=0 \sim M_n-1$) or $k[j]$ ($j=0 \sim M_k-1$). To represent Euclidean distance recognizers and LSS recognizers with different n , we use $k[0] = -1$ ($v^k_1 = -1$) for the LSS recognizers and $k[1] = 0$ ($v^k_2 = 0$) for the Euclidean distance recognizers.

We design a linear structure for constructing a coarse classifier, as shown in Fig. 4.

Figure 4. Linear structure for constructing a coarse classifier. Five stages (nodes) denoted by N_i ($i=1\sim5$). Each node has a basic recognizer.

We assume five stages (nodes) denoted by N_i ($i=1\sim5$). Each node has a basic recognizer, where the parameter n is set as $n[I_i^n]$ ($I_i^n=0\sim M_n-1$) and k is set as $k[I_i^k]$ ($I_i^k=0\sim M_k-1$). Each recognizer of each node N_i has o_{i-1} input candidates and o_i output candidates, where o_0 is the number of all character classes. The term u_i ($i=1\sim5$) denotes if the node N_i is used so that N_i is not used when $u_i = 0$; otherwise, it is used. By this formulation, we can find the best cascade within five stages rather than just five fixed stages.

For the coarse classifier, an input feature vector is input to each node in order from N_1 to N_5 . For each node N_i , the input feature vector is reduced from 517 to $n[I_i^n]$, then the basic recognizer of N_i uses the $n[I_i^n]$ features to search the o_{i-1} input candidates and compare them, then it selects the o_i top best candidates from the o_{i-1} input candidates as its output candidates. Then, the o_i output candidates are set as the input candidates of the next N_{i+1} . If $u_i = 0$ and N_i is not used, is skipped, and its o_{i-1} input candidates are set as the input candidates of the next node N_{i+1} . Finally, the coarse classifier outputs a set of candidates, and these candidates are input into the fine classifier.

Each node N_i has a set of parameters $\{I_i^n, I_i^k, o_i, u_i\}$ whose value ranges are taken as an integral number, as shown in Table 3.

Table 3. Value ranges of parameters

Node Parameters	N_1	N_2	N_3	N_4	N_5
I_i^n	$0\sim M_n-1$	$0\sim M_n-1$	$0\sim M_n-1$	$0\sim M_n-1$	$0\sim M_n-1$
I_i^k	$0\sim M_k-1$	$1\sim M_k-1$	$1\sim M_k-1$	$1\sim M_k-1$	$1\sim M_k-1$
o_i	$1\sim o_0$	$1\sim o_{m_2}$	$1\sim o_{m_3}$	$1\sim o_{m_4}$	$1\sim o_{m_5}$
u_i	1	0 or 1	0 or 1	0 or 1	0 or 1

Here, m_i ($i=2\sim5$) in the third row of Table 3 denotes the number of the previous nearest used node, and o_{m_i} is the

number of input candidates of N_i . For example, if u_2 is 1 and u_3 is 0, then N_2 is used and N_3 is not used so that m_4 is 2 and o_4 can be taken from 1 to o_2 .

If $k[I_i^k] > n[I_i^n]$ for a node N_i , I_i^k is set as $\text{mod}(I_i^k, \text{max_id_k}(I_i^n)+1)$, where max_id_k is a function that takes the index for $k[M_k]$ of the maximum k , which the MQDF recognizer with $n[I_i^n]$ can take, and mod is a residue function that takes the residue of $I_i^k / (\text{max_id_k}(I_i^n)+1)$.

LSS recognizers are for a large input candidate set, and their input candidates have to be fixed so that the LSS recognizers can be used only at the first node N_1 . Therefore, only I_1^k can be taken as 0. Moreover, to guarantee at least one node is used, u_i is fixed as 1.

The five sets of parameters $\{I_i^n, I_i^k, o_i, u_i\}$ ($i=1\sim5$) are estimated and trained by using GA so as to optimize the performance. When training the parameters with GA, the values of the parameters are taken automatically according to the value ranges shown in Table 1.

4. Parameter Optimization

GA optimizes the following objective speed score Score_t .

$$\begin{aligned} \text{Score}_t &= (t_{\max} - t_{\text{whole}}) \text{Num_train} \\ t_{\text{whole}} &= t_{\text{coarse}} + o_{\text{coarse}} t_{\text{fine}}^{\text{average}} + t_{\text{search}}^{o_{\text{coarse}} - o_{\text{fine}}} \\ t_{\text{coarse}} &= \sum_{i=1}^5 (o_{m_i} t_i^{\text{average}} + t_{\text{search}}^{o_{m_i} - o_i}) I(u_i = 1) \end{aligned} \quad (2)$$

where $T_{\max}$ is a very larger constant, t_{whole} is the recognition time to recognize a character, Num_train is the number of training data, t_{coarse} is the recognition time for the coarse classifier, o_{coarse} is the number of candidates output by the coarse classifier, $t_{\text{fine}}^{\text{average}}$ is the average time to calculate the score of the input pattern with a candidate class for the fine classifier, o_{fine} is the number of candidates output by the fine classifier, $t_{\text{search}}^{o_{\text{coarse}} - o_{\text{fine}}}$ is the search time to select the o_{fine} top

best candidates from the o_{coarse} candidates, $I(\cdot)$ is an indicator function that takes value 1 when the condition in the parentheses is satisfied, otherwise it takes value 0, $t_i^{average}$ is the average time to calculate the score of the input pattern with a candidate class for the recognizer decided by $n[I_i^n]$, and $k[I_i^k]$, $t_{search}^{o_{m_i}-o_i}$ is the search time to select the o_i top candidates from the o_{m_i} candidates. If the condition is not satisfied that the cumulative rate of the output candidates of the coarse classifier is more than 99.9%, $Score_t$ is set as 0.

We use a hash sort method to sort the scores of input candidates and select the top best output candidates so that the search time to select top candidates is almost 0. In our experiment, we have found that, for a large set of input and output candidates, using simple sort methods takes a very large amount of processing time.

We treat each one of the $\{I_i^n, I_i^k, o_i, u_i\}$ ($i=1\sim 5$) as an element of a chromosome. A chromosome has 20 elements. The parameters are estimated by GA in the following steps.

- (1) Initialization: Initialize O chromosomes with random values between the value ranges of each parameter shown in Table 3 and the average fitness of the O chromosomes f_{old} as 0 and time t as 1.
- (2) Crossover: Select two chromosomes at random from the O chromosomes. Cross the elements between two random positions to produce two new chromosomes. Repeat until obtaining the M new chromosomes.
- (3) Mutation: Change each element of the $O+M$ chromosomes with a random value between the value ranges of each parameter shown in Table 3 at a probability P_{mut} .
- (4) Fitness evaluation: Evaluate fitness in terms of the speed score in (2) 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 by using the roulette until O new chromosomes are obtained. Replace the old O chromosomes with the new ones.
- (6) Iteration: Obtain the average fitness of the new O chromosomes f_{new} . If $(f_{new} - f_{old} < \text{threshold})$ occurs n_{stop} times or $t > T$, return the chromosome of the highest fitness. Otherwise, set f_{new} to f_{old} , increase t , and go to step 2.

5. Experiments

We take M_n as 21, M_k as 13, and each v_i^n ($i=1 \sim M_n$), v_j^n ($j=1 \sim M_k$) as follows.

$$n[21] = \{01, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200\}$$

$$k[13] = \{-1, 0, 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$$

We can also use other values for n and k , but for the convenience of our implementation, we use only these values. Therefore, we can obtain 243 basic recognizers (201 MQDF recognizers, 21 Euclidean distance recognizers, and 21 LSS recognizers). For the parameter optimization by GA, We set O as 10, M as 10, P_{mut} as 0.03, n_{stop} as 25, and T as 10,000.

An iteration of GA will evaluate many sets of parameters, and we have to recognize all training data by the coarse classifier every time to evaluate a set of parameters. This will take a long time, and it is very difficult to obtain the training result. Therefore, to save computation, for each training sample, we use all basic recognizers (243 basic recognizers) to recognize it and select hundreds of top best candidates for each basic recognizer and store them into a file before applying GA. We also test the average time to calculate the score of the input pattern with a candidate class for each basic recognizer and the fine classifier and then store them into a file. When applying GA to train the parameters, we use the stored candidates to decide the output candidates for each training sample and use the stored time values to evaluate the speed scores.

To evaluate the coarse classifier, we trained 243 basic recognizers, three fine recognizers (a structural MRF recognizer, a structural HMM recognizer and an un-structural MQDF recognizer), and the parameters of the linear structure for constructing the coarse classifier for each database by using the training data. The performance test was made on the testing data. The training and testing data as well as the environment of the experiments (CPU and memory size) are the same as those in the experiments described in Section 3.2.

The speed rate between two different parameter combinations (two different cascades of basic recognizers) is not changed between a PC of high performance and that of low performance in our experiments. Employing a PC of high performance is effective to find the best cascade architecture and it does not change the experimental results for parameter optimization according to the formula of the objective speed score in (1).

For each fine recognizer, we compared the performance of three coarse classifiers: the proposed classifier in this paper, a LSS with Euclidean distance classifier [13-15], and an Euclidean distance classifier. The LSS with a Euclidean distance classifiers applied the optimal value $n=160$ to extract features, used these features to create the LSS recognizers and the

Euclidean distance recognizers, then set the LSS recognizers as the top layer classifiers and the Euclidean distance recognizers as the second layer classifiers. The Euclidean distance coarse classifiers used the same Euclidean distance recognizers without the LSS stages. For the two types of coarse classifiers, we tried two numbers of the output top candidates: 200 and 40. We also compared the performances without the coarse classifiers.

We measured character recognition rate C_r , average character feature extraction time for coarse classification T_{fea_coarse} , average character coarse classification time T_{coarse} , average character feature extraction time for fine classification T_{fea_fine} , average character fine classification time T_{finer} , and average character whole processing time T_{whole} . Table 4 shows the trained parameters obtained by GA. Table 5-6 show the results for each fine recognizer, where *cand* denotes the number of the output top candidates.

Table 4. Trained parameters obtained by GA

MRF fine recognizer	HMM fine recognizer	MQDF fine recognizer
$\{p_1, k_1, o_1, u_1\} = \{08, 01, 84, 1\}$	$\{p_1, k_1, o_1, u_1\} = \{08, 01, 82, 1\}$	$\{p_1, k_1, o_1, u_1\} = \{18, 01, 78, 1\}$
$\{p_2, k_2, o_2, u_2\} = \{20, 06, 06, 1\}$	$\{p_2, k_2, o_2, u_2\} = \{20, 06, 07, 1\}$	$\{p_2, k_2, o_2, u_2\} = \{14, 04, 17, 1\}$
$\{p_3, k_3, o_3, u_3\} = \{18, 05, 01, 0\}$	$\{p_3, k_3, o_3, u_3\} = \{00, 00, 01, 0\}$	$\{p_3, k_3, o_3, u_3\} = \{00, 05, 01, 0\}$
$\{p_4, k_4, o_4, u_4\} = \{02, 03, 01, 0\}$	$\{p_4, k_4, o_4, u_4\} = \{17, 06, 01, 0\}$	$\{p_4, k_4, o_4, u_4\} = \{02, 03, 01, 0\}$
$\{p_5, k_5, o_5, u_5\} = \{14, 02, 01, 0\}$	$\{p_5, k_5, o_5, u_5\} = \{04, 05, 01, 0\}$	$\{p_5, k_5, o_5, u_5\} = \{16, 01, 01, 0\}$
$\{p_1, k_1, o_1, u_1\} = \{14, 01, 80, 1\}$	$\{p_1, k_1, o_1, u_1\} = \{14, 01, 82, 1\}$	$\{p_1, k_1, o_1, u_1\} = \{08, 01, 80, 1\}$
$\{p_2, k_2, o_2, u_2\} = \{06, 06, 15, 1\}$	$\{p_2, k_2, o_2, u_2\} = \{06, 06, 15, 1\}$	$\{p_2, k_2, o_2, u_2\} = \{14, 02, 25, 1\}$
$\{p_3, k_3, o_3, u_3\} = \{20, 04, 01, 0\}$	$\{p_3, k_3, o_3, u_3\} = \{00, 00, 01, 0\}$	$\{p_3, k_3, o_3, u_3\} = \{00, 06, 01, 0\}$
$\{p_4, k_4, o_4, u_4\} = \{02, 02, 01, 0\}$	$\{p_4, k_4, o_4, u_4\} = \{20, 06, 01, 0\}$	$\{p_4, k_4, o_4, u_4\} = \{02, 06, 01, 0\}$
$\{p_5, k_5, o_5, u_5\} = \{20, 02, 01, 0\}$	$\{p_5, k_5, o_5, u_5\} = \{04, 06, 01, 0\}$	$\{p_5, k_5, o_5, u_5\} = \{20, 01, 01, 0\}$

Upper: CASIA-OLHWDB1.1; lower: TUAT Nakayosi/Kuchibue.

For character recognition rate C_r , we made a significance test (a sign test) for the difference between our method and each of other methods with the result that almost the differences are validated with $p < 0.05$ except 4 methods.

Table 5. Comparison of coarse classifiers for the structural MRF fine recognizers

Performance \ Method		Our method	LSS with Euclidean		Euclidean		Without CC
			<i>Cand</i> =40	<i>Cand</i> =200	<i>Cand</i> =40	<i>Cand</i> =200	
<i>Cr</i> (%)		91.64	91.00 <i>p</i> <0.05	90.81 <i>p</i> <0.05	91.02 <i>p</i> <0.05	90.87 <i>p</i> <0.05	89.42 <i>p</i> <0.05
<i>Time</i> (ms)	<i>T</i> _{fea_coarse}	0.50	0.51	0.50	0.51	0.50	0
	<i>T</i> _{coarse}	0.54	0.42	0.41	0.81	0.82	0
	<i>T</i> _{fea_fine}	0.00	0.00	0.00	0.00	0.00	0.00
	<i>T</i> _{finer}	0.44	2.96	14.89	3.05	14.90	253.40
	<i>T</i> _{whole}	1.49	3.93	15.84	4.37	16.22	253.40
<i>Cr</i> (%)		93.88	93.38 <i>p</i> <0.05	93.55 <i>p</i> <0.05	93.55 <i>p</i> <0.05	93.67 <i>p</i> <0.05	92.75 <i>p</i> <0.05
<i>Time</i> (ms)	<i>T</i> _{fea_coarse}	0.48	0.48	0.49	0.48	0.49	0
	<i>T</i> _{coarse}	0.41	0.44	0.45	0.91	0.91	0
	<i>T</i> _{fea_fine}	0.00	0.00	0.00	0.00	0.00	0.00
	<i>T</i> _{finer}	0.32	0.83	4.12	0.82	4.13	81.24
	<i>T</i> _{whole}	1.25	1.79	5.11	2.24	5.58	81.24

Upper: CASIA-OLHWDB1.1; lower: TUAT Nakayosi/Kuchibue.

Table 6. Comparison of coarse classifiers for the structural HMM fine recognizer

Performance \ Method		Our method	LSS with Euclidean		Euclidean		Without CC
			<i>Cand</i> =40	<i>Cand</i> =200	<i>Cand</i> =40	<i>Cand</i> =200	
<i>Cr</i> (%)		90.44	90.00 p<0.05	89.71 p<0.05	90.02 p<0.05	89.86 p<0.05	88.32 p<0.05
<i>Time</i> (ms)	<i>T</i> _{fea_coarse}	0.50	0.50	0.49	0.50	0.50	0
	<i>T</i> _{coarse}	0.53	0.43	0.42	0.83	0.82	0
	<i>T</i> _{fea_fine}	0.00	0.00	0.00	0.00	0.00	0.00
	<i>T</i> _{finer}	0.45	2.95	14.79	3.00	14.70	253.30
	<i>T</i> _{whole}	1.48	3.90	15.75	4.33	16.02	253.30
<i>Cr</i> (%)		92.56	92.07 p<0.05	92.22 p<0.05	92.25 p<0.05	92.37 p<0.05	91.78 p<0.05
<i>Time</i> (ms)	<i>T</i> _{fea_coarse}	0.47	0.48	0.47	0.48	0.49	0
	<i>T</i> _{coarse}	0.42	0.44	0.44	0.90	0.91	0
	<i>T</i> _{fea_fine}	0.00	0.00	0.00	0.00	0.00	0.00
	<i>T</i> _{finer}	0.33	0.81	4.13	0.83	4.12	81.22
	<i>T</i> _{whole}	1.26	1.78	5.10	2.24	5.56	81.22

Upper: CASIA-OLHWDB1.1; lower: TUAT Nakayosi/Kuchibue.

Table 7. Comparison of coarse classifiers for the un-structural MQDF fine recognizer

Performance \ Method		Our method	LSS with Euclidean		Euclidean		Without CC
			Cand=40	Cand=200	Cand=40	Cand=200	
Cr (%)		92.13	92.07 $p<0.05$	92.10 $p<0.05$	92.09 $p<0.05$	92.13* $p>0.05$	92.14* $p>0.05$
$Time(ms)$	T_{fea_coarse}	0.48	0.50	0.50	0.50	0.51	0
	T_{coarse}	0.56	0.43	0.43	0.79	0.81	0
	T_{fea_fine}	0.20	0.20	0.20	0.20	0.20	0.50
	T_{finer}	0.16	0.36	1.74	0.35	1.73	25.84
	T_{whole}	1.40	1.49	2.87	1.84	3.05	26.34
Cr (%)		94.65	94.46 $p<0.05$	94.53 $p<0.05$	94.60 $p<0.05$	94.64* $p>0.05$	94.67* $p>0.05$
$Time(ms)$	T_{fea_coarse}	0.48	0.48	0.49	0.48	0.49	0
	T_{coarse}	0.31	0.44	0.45	0.91	0.91	0
	T_{fea_fine}	0.20	0.20	0.20	0.20	0.20	0.50
	T_{finer}	0.23	0.40	2.17	0.42	2.18	40.73
	T_{whole}	1.30	1.50	3.31	2.05	3.92	41.50

Upper: CASIA-OLHWDB1.1; lower: TUAT Nakayosi/Kuchibue. The difference of Cr against our method is validated with $p<0.05$ except those marked *.

From the results, we can see:

- (1) From the parameters of Table 4, we can see that the nodes N_i ($i=3\sim5$) were not used and two basic recognizers were selected to construct the coarse classifiers for all fine recognizers. The MQDF fine recognizers resulted in the more candidates' outputs produced by the coarse classifiers because the MQDF fine recognizers have faster recognition speeds. The MRF/HMM fine recognizers for the CASIA-OLHWDB1.1 database resulted in a few candidates' outputs produced by the coarse classifiers, because the MRF/HMM fine recognizers for the CASIA-OLHWDB1.1 database needed more models for each class to maintain high recognition performances and it caused slower recognition speeds.
- (2) The proposed coarse classifiers remarkably improved the character recognition speeds. For the CASIA-OLHWDB1.1 database, the whole recognition time was reduced to 9.4% (MRF fine recognizer), 9.4% (HMM fine recognizer) and 48.8% (MQDF fine recognizer) compared with the LSS with the Euclidean distance classifier, respectively, and it

was reduced to 9.2% (MRF fine recognizer), 9.2% (HMM fine recognizer) and 45.9% (MQDF fine recognizer) compared with the Euclidean distance coarse classifier, respectively, while the numbers of the output top candidates were taken as 200 to maintain high classification and recognition rates. For the Nakayosi/Kuchibue database, the whole recognition times was reduced to 24.5% (MRF fine recognizer), 24.7% (HMM fine recognizer) and 39.3% (MQDF fine recognizer) compared with the LSS with the Euclidean distance classifier, respectively, and it was reduced to 22.4% (MRF fine recognizer), 22.7% (HMM fine recognizer) and 33.2% (MQDF fine recognizer) compared with the Euclidean distance coarse classifier, respectively, while the numbers of the output top candidates were taken as 200.

- (3) Recognitions without coarse classifiers brought a larger recognition costs.
- (4) The LSSs reduced the numbers of candidates from 3,755 to 1,371 for the CASIA-OLHWDB1.1 database and from 4,438 to 986 for the Nakayosi/Kuchibue database, respectively, and the LSS with the Euclidean distance classifiers were more effective than the Euclidean distance classifiers except in the cases of the MRF and HMM fine recognizers for the CASIA-OLHWDB1.1 database.
- (5) The LSS with the Euclidean distance classifiers and the Euclidean distance classifiers were not effective when the candidates' outputs by the Euclidean distance recognizers were larger in order to guarantee high recognition accuracies.
- (6) The MRF fine recognizers exhibited higher recognition accuracies than the HMM fine recognizers, and the MQDF fine recognizers resulted in the highest recognition accuracies.
- (7) The MRF fine recognizers and the HMM fine recognizers took about the same recognition times. The MQDF fine recognizers resulted in the smallest character fine classification time T_{finer} .
- (8) The MQDF fine recognizers used the same 512 un-structural features extracted from each on-line character pattern as the coarse classifiers, so that they did not need to consume any time on extracting the 512 un-structural features and only consume some time to reduce the 512 features to n features by the FLDA, while the structural feature extraction time for the MRF fine recognizers and that for the HMM fine recognizers are almost 0.
- (9) For our method, the coarse classifier memory sizes of the MRF fine recognizers and those of the HMM fine recognizers were the same (about 30MB for the CASIA-OLHWDB1.1 database, about 12MB for the Nakayosi/Kuchibue database), and those of the MQDF fine recognizers were about 12MB (CASIA-OLHWDB1.1) and about 3MB (Nakayosi/Kuchibue). The fine recognizer memory sizes of MRFs and those of HMMs were the same

(about 42MB for the CASIA-OLHWDB1.1 database, about 12MB for the Nakayosi/Kuchibue database), and those of MQDFs were about 59MB for the CASIA-OLHWDB1.1 database, and about 70MB for the Nakayosi/Kuchibue database.

- (10) The MRF/HMM fine recognizers for the CASIA-OLHWDB1.1 database needed more models for each class to maintain high recognition performances resulting in slower recognition speeds and larger memory costs.
- (11) The fine recognizers (MRF, HMM and MQDF) for the Nakayosi/Kuchibue database exhibited higher recognition accuracies than those for the CASIA-OLHWDB1.1 database. We consider that it is because Chinese characters have simpler shapes than Japanese characters and Chinese people writes characters more fluently and more cursively than Japanese people resulting in low recognition accuracies.

The resultant coarse classifier makes the whole recognition composed of itself, fine classification, a segmentation module, geometric context processor and linguistic context processing fast enough even on a small devices. In fact, it is employed in those devices and the whole recognition speed is about 0.3 seconds per character for the iPads (A5 Dual Core CPU 1GHz with 512MB memory) and the smart phones (Samsung Exynos 4210 Dual-core CPU 1.4GHz with 1GB memory) .

6. Conclusion

In this paper, a systematic method for constructing an efficient and robust coarse classifier by using GA for the on-line recognition of handwritten Chinese and Japanese characters was presented. We created 243 basic recognizers with different classification costs and different classification accuracies. We made the coarse classifier as their sequential cascade, which reduced candidates one after another. The parameters were estimated by using GA. The whole recognition time was reduced largely from the original, by only the two stages of the basic recognizers while recognition rates were maintained.

Acknowledgment

This work is partially supported by the R&D fund for the "Development of pen & paper based user interaction" under

the Japan Science and Technology Agency.

References

- [1] B. Zhu, X.-D. Zhou, C.-L. Liu and M. Nakagawa: A robust model for on-line handwritten Japanese text recognition, *International Journal on Document Analysis and Recognition (IJ DAR)*, Vol. 13, No. 2, pp.121-131, 2010.
- [2] 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, 2011.
- [3] C.-L. Liu and X.-D. Zhou: Online Japanese character recognition using trajectory-based normalization and direction feature extraction, *Proc. 10th International Workshop on Frontiers in Handwriting Recognition*, pp.217-222, 2006.
- [4] B. Zhu and M. Nakagawa: A MRF Model with parameters optimization by CRF for on-line recognition of handwritten Japanese characters, *Proc. Document Recognition and Retrieval XVIII (DRR) that is part of IS&T/SPIE Electronic Imaging*, San Jose, USA, 2011.
- [5] B. Zhu and M. Nakagawa: On-line handwritten Japanese characters recognition using a MRF model with parameter optimization by CRF, *Proc. 11th ICDAR*, pp. 603-607, Beijing, China, 2011.
- [6] S. Mori, K. Yamamoto and M. Yasuda: Research on machine recognition of handprinted characters, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.6, pp.386-405, 1984.
- [7] Hildebrandt, T. H., Liu, W. T., Optical recognition of handwritten chinese characters: Advances since 1980. *Pattern Recognition* 26 (2), 05–225, 1993.
- [8] Wakabayashi, T., Deng, Y., Tsuruoka, S., Kimura, F., Miyake, Y., Accuracy improvement by nonlinear normalization and feature compression in handwritten chinese character recognition. *IEICE PRU* 95 (43), 1–8, 1995.
- [9] Sun, N., Abe, M., Nemoto, Y., A handwritten character recognition system by using improved directional element feature and subspace method. *J. IEICE* 78 (6), 922–930, 1995.
- [10] C.-L. Liu and M. Nakagawa: Precise candidate selection for large character set recognition by confidence evaluation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22 (6), 636-642, 2000.
- [11] Kumamoto, T., Toraichi, K., Horiuchi, T., Yamamoto, K., Yamada, H., On speeding candidate selection in handprinted chinese character recognition. *Pattern Recognition* 24 (8), 793–799, 1991.
- [12] Tung, C.-H., Lee, H.-J., Tsai, J.-Y., Multi-stage pre-candidate selection in handwritten chinese character recognition systems. *Pattern Recognition* 27 (8), 1093–1102, 1994.
- [13] Y. Yang and M. Nakagawa: Accelerating large character set recognition using pivots, *Proc. 8th International Conference Document Analysis and Recognition*, Edinburgh, pp.262-267, 2003.
- [14] Y. Yang, B. Zhu and M. Nakagawa: Structuring search space for accelerating large set character recognition, *IEICE Trans.*

Information and Systems, E88-D (8), pp.1799-1806, 2005.

- [15] Y. Yang and M. Nakagawa: Improving the structuring search space method for accelerating large set character recognition, Proc. 9th International Workshop on Frontiers in Handwriting Recognition, Tokyo, 251-256, 2004.
- [16] Tseng, Y. H., Kuo, C. C., Lee, H. J., Nov. 1998. Speeding up Chinese character recognition in an automatic document reading system. Pattern Recognition 31 (11), 1601–1612, 1998.
- [17] Fujimoto, K., Kamada, H., Kurokawa, K., Fast, precise pre-classification method using projections of feature regions. Technical report of IEICE. PRMU 97 (558), 25–32, 1998.
- [18] Y. Waizumi, N. Kato, K. Saruta and Y. Nemoto: High speed and high accuracy rough classification for handwritten characters using hierarchical learning vector quantization (in Japanese), Technical Report of IEICE, Vol.97, No.40, PRMU, pp.1-8, 1997.
- [19] M. Nakagawa and K. Matsumoto: Collection of on-line handwritten Japanese character pattern databases and their analysis, International Journal on Document Analysis and Recognition (IJ DAR), 7(1), pp.69-81, 2004.
- [20] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang: Online and offline handwritten Chinese character recognition: Benchmarking on new databases, Pattern Recognition, 46(1): 155-162, 2013.
- [21] B. Zhu and M. Nakagawa: A coarse classifier construction method from a large number of basic recognizers for on-line recognition of handwritten Japanese characters, Proc. 11th ICDAR, pp.1090-1094, Beijing, China, 2011.
- [22] F. Kimura: Modified quadratic discriminant function and the application to Chinese characters, IEEE Trans. Pattern Analysis and Machine Intelligence, 9 (1), pp.149-153, 1987.
- [23] C.-L. Liu and K. Marukawa: Pseudo two dimensional shape normalization methods for handwritten Chinese character recognition, Pattern Recognition, 38(12), pp.2242–2255, 2005.
- [24] C.-L. Liu: High accuracy handwritten Chinese character recognition using quadratic classifiers with discriminative feature extraction, Proc. 18th International Conference on Pattern Recognition, vol. 2, Hong Kong, pp.942–945, 2006.
- [25] U. Ramer: An iterative procedure for the polygonal approximation of plan closed curves, Computer Graphics and Image Processing, vol. 1, pp.244-256, 1972.

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.

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. 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.

Figure 1

Figure 2

Start

a

b

c

d

End

Figure 3

(a) Stroke length

$$X = X_1 + X_2 + X_3$$

$$Y = Y_1 + Y_2 + Y_3$$

(b) Direction change times

X direction : once

Y direction: three times

Figure 4