A Coarse Classifier Construction Method from a Large Number of Basic Recognizers for On-line Recognition of Handwritten Japanese Characters

Bilan Zhu and Masaki Nakagawa
Department of Computer and Information Sciences,
Tokyo University of Agriculture and Technology,
Tokyo 184-8588, Japan
{zhubilan, nakagawa}@cc.tuat.ac.jp

***Abstract*—This paper describes a method for constructing the most efficient and robust coarse classifier from a large number of basic recognizers which are obtained by different parameters of feature extraction, different discriminant methods or functions, and so on. The architecture of the coarse classification is a sequential cascade of basic recognizers and reduces the candidates after each basic recognizer. Genetic algorithm determines the best cascade with the best speed and highest performance. The method is applied for on-line handwritten Japanese characters recognition. We produced 201 basic recognizers of MQDF, 21 basic recognizers of Euclidian distance and 21 basic recognizers of the LSS method by changing parameters. From these basic recognizers we have obtained a rather simple 2 stages cascade with the result that the whole recognition time was reduced to 24.5% while keeping classification and recognition rates.**

Keywords-On-Line character recgnition; Japanese character recgnition; Coarse classifier; Genetic algorithm

I. INTRODUCTION

With the development and proliferation of pen-based input devices such as tablet PCs, memo or note pads, electronic whiteboards and digital pens (e.g., Anoto pen), on-line handwritten characters recognition with high recognition speed and high recognition accuracy is in demand. Although character classifiers with high recognition accuracy have been reported [1-4], the demand for speeding up recognition is very high for portable devices as well as desk-top applications for which handwriting recognition is incorporated as one of modules.

Large character set recognition is problematic not only in recognition rate but also in recognition speed. Chinese, Japanese or Korean have thousands of different categories, so that recognition takes more time than Latin alphabets or numerals. A general approach to improve the recognition speed is to perform coarse classification, pre-classification or candidate selection before the fine classification [5, 6].

Candidate selection or coarse classification started early in the history character recognition since the machine power was poor [7, 8], but still it is an essential topic. We have also reported "layered search spaces" (LSS) to accelerate recognition of a large category set [9-11]. However, the improvement for recognition speed is still in great demand.

In this paper, we present a robust coarse classifier construction method by genetic algorithm for on-line recognition of handwritten Japanese characters. The method creates 243 basic recognizers with different classification costs and different classification accuracies by controlling the parameters of feature extraction and discriminant function as well as the layered search spaces (LSS) method. Then it uses these basic recognizers to construct a robust coarse classifier. It constructs a sequential cascade of basic recognizers and reduces the candidates after each basic recognizer. The parameters are estimated by the genetic algorithm so as to optimize the holistic character recognition performance. Experimental results on the TUAT Kuchibue database [12] demonstrate the superiority of our method.

The rest of this paper is organized as follows: Section 2 presents an overview of our on-line handwritten character recognition system. Section 3 designs a linear structure for constructing a coarse classifier and Section 4 describes the parameter optimization method. Section 5 presents the experimental results and Section 6 draws our conclusion.

II. RECOGNITION SYSTEM OVERVIEW

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

On-line recognizer is in fact one of other modules for handwritten text recognition [1] and it is combined with an off-line recognizer [2] to obtain the robustness against stroke disorder as well as a segmentation module, geometric context processor and a linguistic postprocessor.

As for the coarse classification, we will present its design and implementation in the following sections.

Fine classification after coarse classification is based on a

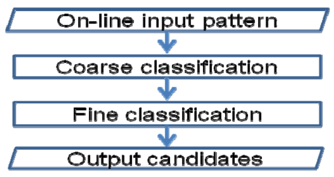

Figure 1. On-line handwritten character recognition.

MRF model [4]. We extract feature points along the pen-tip trace from pen-down to pen-up. We employ the coordinates of feature points as unary features and the differences in coordinates between the neighboring feature points as binary features. Then we use a MRF model to match the feature points with the states of each character class of input candidates 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.

While fine classification in the on-line recognizer should exploit the on-line features since the on-line recognizer is combined with the off-line recognizer, coarse classification in the on-line recognizer can use not only on-line features and classification methods but also those for off-line recognition as well as any other methods such as LSS that does not evaluate the scores of character patterns.

III. Linear Structure Design for Constructing Coarse Classifier

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

A. Preprocessing and Feature Extraction

From on-line character patterns (sequences of stroke coordinates), we extract direction features: histograms of normalized stroke direction [3]. For the coordinate normalization methods we apply pseudo 2D bi-moment normalization (P2DBMN) [13]. The local stroke direction is decomposed into 8 directions and from the feature map of each direction, 8x8 values are extracted by Gaussian blurring. So, the dimensionality of feature vectors is 512.

Moreover, we also extract 6 features from each on-line character pattern which have been effectively applied on our coarse classification step. After the P2DNMN normalization, we extract feature points using the method by Ramner [14]. 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 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 6 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 518 features from an on-line character pattern. Then, to improve the Gaussianity of feature distribution, each value of the 518 features is transformed by Box-Cox transformation (also called variable transformation).

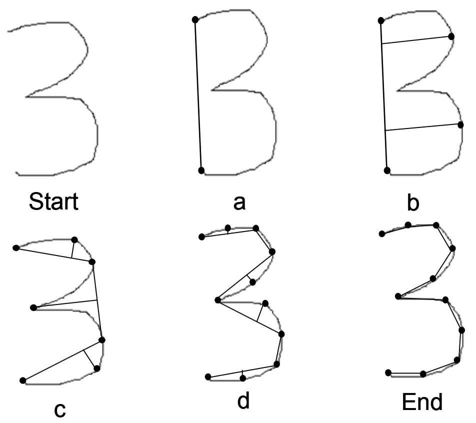

Figure 2. Feature points extraction.

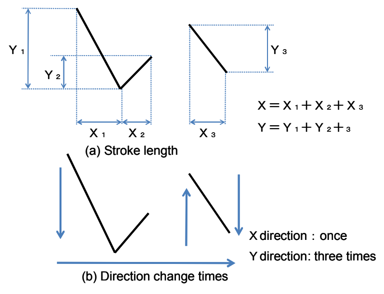

Figure 3. On-line character features.

B. Basic Character Recognizers

The input feature vector is first reduced from 518D to *n*D by Fisher linear discriminant analysis (FLDA). Then we can use the *n*D feature vectors to create modified quadratic discriminant function (MQDF) recognizer [15] as follows:

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

where μ_i is the mean vector of class ω_i, λ_{ij} (j = 1, …, 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 eigen vector which is set as a constant. The value of δ can been optimized on the training data set, however for a convenience we simply set it as $\alpha\lambda_{average}$ where $\lambda_{average}$ is the average of λ_{ij} (i,j = 1, …, n) for all features of all classes and α is a constant that is larger than 0 and smaller than 1.

According to the previous works [3, 16], the best performance is obtained when n is about 160 and k is about 50. When n and k are smaller than their optimal values, although its top recognition rate is degraded, it can recognize an input pattern with higher speed.

Here, we introduce the cumulative recognition rate as the rate that the correct class is listed within the top-$\mathcal{N}$ candidates by the recognizer. For coarse classification, we set $\mathcal{N}$ large so that the sufficiently high cumulative rate is assured and the correct class is passed to the fine classification.

For MQDF recognizers as coarse classifiers with smaller n and k than their optimal values, we set larger $\mathcal{N}$ 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 smaller memory cost and retain the correct candidate within top-$\mathcal{N}$by setting $\mathcal{N}$large enough. Without experiments, we do not know which recognizer is the best for coarse classification.

We have developed "layered search spaces" (LSS) to accelerate recognition of a large category set [9-11]. The basic concept is to employ pivots into a 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 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 LSS we can also use different feature dimensionality n. Under the condition that the top-$\mathcal{N}$ cumulative rate is more than 99.9%, n smaller than the optimal value brings lower processing cost and larger output candidate set while the larger n unless it is larger than the optimal value, brings higher recognition costs and smaller output candidate set.

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

C. *Linear Structure Design*

We create 243 basic recognizers of MQDF and LSS by adjusting the FLDA parameter n and the MQDF parameter 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}

where each value is denoted as $n[l]$ (l =0~20) or $k[j]$ (j =0~12). To represent Euclidean distance recognizers and LSS recognizers with different n, we use k[0] = -1 for LSS recognizers and k[1]= 0 for Euclidean distance recognizers. We can also use other values for n and k, but for the convenience of our implementation we only use these values.

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

We assume five stages (nodes) denoted by N_i (i=1~5). Each node has a basic recognizer where the parameter n is set as $n[I^n_i]$ (I^n_i=0~20) and k is set as $k[I^k_i]$ (I^k_i=0~12). 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

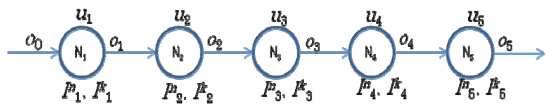

Figure 4. Linear structure for constructing coarse classifier.

classes. The term u_i (i=1~5) 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 5 stages rather than just fixed 5 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 518 to $n[I^n_i]$, then the basic recognizer of N_i uses the $n[I^n_i]$ features to search o_{i-1} input candidates and to compare them, then it selects 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, it 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 MRF fine classifier.

Each node N_i has a set of parameters $\{I^n_i, I^k_i, o_i, u_i\}$ whose value ranges are taken as an integral number as shown in Table 1.

TABLE I. THE VALUE RANGES OF PARAMETERS

Node / Parameters	N_1	N_2	N_3	N_4	N_5
I^n_i	0~20	0~20	0~20	0~20	0~20
I^k_i	0~12	1~12	1~12	1~12	1~12
o_i	1~M	1~ o_{m_2}	1~ o_{m_3}	1~ o_{m_4}	1~ o_{m_5}
u_i	1	0 or 1	0 or 1	0 or 1	0 or 1

where m_i (i =2~5) in the third row in Table 1 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 $k[I^k_i] > n[I^n_i]$ for a node N_i, I^k_i is set as $mod(I^k_i, max\_id\_k(I^n_i)+1)$ where max_id_k is a function that takes the index for $k[13]$ of the maximum k which MQDF recognizer with $n[I^n_i]$ can taken, and mod is a residue function that takes the residue of I^k_i /($max\_id\_k(I^n_i)$+1).

LSS recognizers are for a large input candidate set and their input candidates have to been fixed with the result that LSS recognizers can only been used at the first node N_1. Therefore, only I^k_1 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 $\{I^n_i, I^k_i, o_i, u_i\}$ (i=1~5) are estimated and are trained by GA so as to optimize the performance. When training the parameters by GA the values of parameters are taken automatically according to the value ranges as shown in Table 1.

IV. PARAMETER OPTIMIZATION

GA optimizes the following objective speed score *Score_t*:

$$\begin{aligned} Score\_t &= (t_{\max} - t_{whole})Num\_train \\ t_{whole} &= t_{coarse} + o_{coarse}t^{average}_{fine} + t^{o_{coarse}-o_{fine}}_{search} \\ t_{coarse} &= \sum_{i=1}^{5}(o_{m\_i}t^{average}_{i} + t^{o_{m\_i}-o_i}_{search})I[u_i = 1] \end{aligned} \tag{2}$$

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, $t_{fine}^{average}$ 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, $t_{search}^{o_{coarse}-o_{fine}}$ is the search time to select o_{fine} top best candidates from o_{coarse} candidates, I (·) is an indicator function which takes value 1 when the condition in the parentheses is satisfied, otherwise 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 $n[I^n_i]$ and $k[I^k_i]$, $t_{search}^{o_{m\_i}-o_i}$ is the search time to select o_i top candidates from o_{m_i} candidates. If the condition is not satisfied that the cumulative recognition correct 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 top best output candidates with the result that the search time to select top candidates is almost 0. In our experiment we have found for a large set of input candidates and output candidates, using simple sort methods takes very large processing time.

An iteration of GA will evaluate many sets of parameters, and we have to recognize all training data by the coarse classifier everytime to evaluate a set of parameters. This will take large 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.

We treat each one of $\{I^n_i, I^k_i, o_i, u_i\}$ (i=1~5) as an element of a chromosome. Chromosome has 20 elements. The parameters are estimated by GA in following steps:

(1) Initialization: Initialize N chromosomes with random values between the value ranges of each parameter shown in Table 1, average fitness of the N chromosomes f_{old} as 0 and time t as 1.

(2) Crossover: Select two chromosomes at random from N chromosomes. Cross the elements between two random positions to produce two new chromosomes. Repeat until obtaining M new chromosomes.

(3) Mutation: Change each element of N+M chromosomes with a random value between the value ranges of each parameter shown in Table 1 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 using the roulette until obtaining N new chromosomes. Replace the old N chromosomes with the new ones.

(6) Iteration: Obtain the average fitness of the new N chromosomes f_{new}. If (f_{new} - f_{old} < threshold) occurs n_{stop} times or t > T, return the chromosome of the highest fitness. Otherwise, set f_{new} to f_{old}, increment t, and go to step 2.

We set N as 10, M as 10, P_{mut} as 0.03, n_{stop} is as 25 and T as 10,000.

V. Experiments

To evaluate the coarse classifier, we trained 243 basic recognizers (201 MQDF recognizers, 21 Euclidean distance recognizers and 21 LSS recognizers), the character recognizer of the MRFs (fine classifier) and the parameters of the linear structure for constructing the coarse classifier by using an on-line Japanese handwriting database called Nakayosi [12]. On the other hand, the performance test was made on an on-line Japanese handwriting database called Kuchibue [12]. Table 2 shows the details of the databases. Each character class (character category) has a different number of sample patterns, and kana and symbol 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 processers) with 12 GB memory.

TABLE II. Statistics of character pattern databases.

		Nakayosi_t	Kuchibue_d
#writers		163	120
#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

We compared the performance of three coarse classifiers: the proposed classifier in this paper, a LSS with Euclidean distance classifier [9-11] and an Euclidean distance classifier. The previous works always applied the optimal values n such as 160 to extract features and to create the coarse classifiers [3, 9-11]. Therefore, a LSS with a Euclidean distance classifier applies the optimal values n=160 to extract features and uses these features to create the LSS recognizer and the Euclidean distance recognizer, then sets the LSS recognizer as the top layer classifier and the Euclidean distance recognizer as the second layer classifier. The Euclidean distance coarse classifier uses the same Euclidean distance recognizer without the LSS stage. For the two coarse classifiers we try two numbers of the output top candidates: 200 and 40. We also compared the performance without the coarse classifier.

We use character recognition rate C_r, average character feature extraction time T_{fea}, average character coarse classification time T_{coarse}, average character fine classification time T_{finer}, and average character whole processing time T_{whole}. Table 3 shows the results where *cand* denotes the number of the output top candidates. For reference, the trained parameters obtained by GA are as follows:

$\{I^n_1, I^k_1, o_1, u_1\}=\{14,01,80,1\}$
$\{I^n_2, I^k_2, o_2, u_2\}=\{06,06,40,1\}$
$\{I^n_3, I^k_3, o_3, u_3\}=\{21,04,01,0\}$
$\{I^n_4, I^k_4, o_4, u_4\}=\{02,02,01,0\}$
$\{I^n_5, I^k_5, o_5, u_5\}=\{20,02,01,0\}$

From the parameters, we can see that the nodes N_i (i=3~5) are not used and two basic recognizers are selected to construct the coarse classifier.

TABLE III. COMPARISON OF COARSE CLASSIFIERS.

Method / Performance		Our method	LSS with Euclidean		Euclidean		Without CC
			Cand=40	*Cand*=200	*Cand*=40	*Cand*=200	
Cr (%)		93.88	93.38	93.55	93.55	93.67	92.75
Time(ms)	T_{fea}	0.48	0.48	0.49	0.48	0.49	0
	T_{coarse}	0.41	0.44	0.45	0.91	0.91	0
	T_{finer}	0.32	0.83	4.12	0.82	4.13	81.24
	T_{whole}	1.25	1.79	5.11	2.24	5.58	81.24

From the results, we can see that the proposed coarse classifier remarkably improve the character recognition speed; the whole recognition time is reduced to 24.5% compared to the LSS with Euclidean distance classifier, 22.4% compared to the Euclidean distance coarse classifier, respectively, while the numbers of the output top candidates are taken as 200 to keep high classification and recognition rates. We can see that recognition without a coarse classifier brings larger recognition cost. The LSS reduces the number of candidates from 4,438 to 986, when the recognition time of the fine classifier is small and the number of the candidates output by the Euclidean distance recognizer is small such as 40, the LSS with the Euclidean distance classifier is more effective than the Euclidean distance classifier. However, when the recognition time of the fine classifier is larger and the candidates output by the Euclidean distance recognizer is larger to guarantee high recognition accuracy, the LSS with the Euclidean distance classifier is not effective.

VI. CONCLUSION

This paper presented a robust coarse classifier construction method by GA for on-line recognition of handwritten Japanese characters. 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 a genetic algorithm. The whole recognition time is reduced to 24.5%, that is about 1/4 of the original while keeping recognition rates only by the two stages of basic recognizers.

ACKNOWLEDGMENT

This work is being partially supported by the R&D fund for "development of pen & paper based user interaction" under 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 (IJDAR),Vol. 13, No. 2, pp.121-131, 2010.

[2] H. Oda, B. Zhu, J. Tokuno, M. Onuma, A. Kitadai and M. Nakagawa: A compact on-line and off-line combined recognizer. Proc. 10th International Workshop on Frontiers in Handwriting Recognition, pp. 133–138. La Baule, France, 2006.

[3] C.-L. Liu and X.-D. Zhou: Online Japanese caracter 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] 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.

[6] 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.

[7] 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.

[8] T. H. Hildebrandt and W. Liu: Optical recognition of handwritten Chinese characters: Advances since 1980, Pattern Recognition, vol.26, no.2, pp.205-225, 1993.

[9] Y. Yang and M. Nakagawa: Acclerating large character set recognition using pivots, Proc. 8th International Conference Document Analysis and Recognition, Edinburgh, pp.262-267, 2003.

[10] 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.

[11] 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.

[12] M. Nakagawa and K. Matsumoto: Collection of on-line handwritten Japanese character pattern databases and their analysis, International Journal on Document Analysis and Recognition (IJDAR), 7(1), pp.69-81, 2004.

[13] 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.

[14] U. Ramer: An iterative procedure for the polygonal approximation of plan closed curves, Computer Graphics and Image Processing, vol. 1, pp.244-256, 1972.

[15] 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.

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.