Fast and Accurate Tree-Based Clustering for Japanese/Chinese Character Recognition

Yuichi Abe[1], Takahiro Sasaki[1], and Hideaki Goto[2]

[1] Graduate School of Information Sciences,
Tohoku University, Japan 980-8578
[2] Cyberscience Center, Tohoku University, Japan 980-8578

Abstract. Recognizing text in natural scene images is very important to develop various systems such as an assistant device for visually-impaired people. Multilingual scene text recognition is also becoming important for wearable camera devices with language translation feature. Since computational resources are limited on such mobile devices, fast and accurate Optical Character Recognition (OCR) algorithm is needed. Nearest Neighbor (NN) search is quite popular in feature vector-based OCR systems, and its speed improvement is required. In this paper, we develop an OCR scheme with tree-based clustering technique with LDA (Linear Discriminant Analysis) aiming at real-time Japanese/Chinese character recognition. The experimental results using ETL9B dataset show that our proposed method is 94.6% faster than our previous method, also beating other techniques, at mere 0.24% accuracy drop from the full linear search.

Keywords: Fast Nearest Neighbor search, Linear Discriminant Analysis (LDA), real-time character recognition, Approximate Nearest Neighbor (ANN) search, multilingual OCR.

1 Introduction

Scene text recognition is quite useful for developing assistive devices for visually-impaired people, automatic translator for tourists, etc.[1,2]. In such systems, recognizing scene text requires some processes (detecting text regions from scene images, segmenting text, recognizing text), and each of them requires a lot of computation. Some text extraction algorithms utilize feedback from character recognition as proposed in [2], in order to detect/segment character images precisely. There are thousands of characters in Japanese and Chinese, and the character recognition takes much longer time than European languages. In addition, the feature vectors for Japanese/Chinese OCR are in higher dimensionality than alphabet ones because of the characters' complicated shapes. Since a Kanji/Hanzi character can be divided into some connected components, multiple hypotheses testing is often used in the character segmentation stage, and it requires much more trials in character recognition. To achieve a video-rate processing, several hundreds of character candidates per video frame need to be processed, while a

A. Petrosino (Ed.): ICIAP 2013, Part II, LNCS 8157, pp. 459–468, 2013.

conventional rough classification methods such as those based on dimensionality reduction cannot fulfill the demand even on a desktop computer.

Nearest Neighbor (NN) search has been quite popular and still wideliy used in OCR. Approximated Nearest Neighbor (ANN) search [3] is one of the most popular methods for improving the speed of the NN search. However, the degree of its speed improvement is not satisfactory when we try to keep the accuracy drop small in the character recognition tasks. Although the speed acceleration of Locality Sensitive Hash (LSH) scheme [4] is attractive, LSH introduces some accuracy drop when it is directly applied to character recognition.

Sobu and Goto proposed a fast and accurate tree-based method [5]. However the method was only tested on some artificial data. To improve the performance with handwritten characters, we have developed an improved version of Sobu's, and Sasaki has presented it in a technical report [7] (written in Japanese, without peer review).

In this paper, we introduce the Sasaki's method [7], a tree-based clustering technique for accelerating the NN search with high-dimensional feature vectors used in Japanese/Chinese character recognition, minimizing the accuracy drop. We employ multiclass Linear Discriminant Analysis (LDA), which can determine the dividing hyperplanes considering the distribution of each class. In addition, we investigate the performance with Chinese handwritten characters, and seek for a possibility for real-time multilingual OCR. In the experiments, we used ETL9B and HCL2000 hand-written character datasets [6].

The rest of this paper is organized as follows. The definition of the OCR we are using, and some conventional speed-up techniques are briefly given in Section 2. In Section 3, we propose the tree-based clustering and the fast character recognition algorithm. Section 4 describes the experimental results and discussions, and Section 5 concludes this paper.

2 Character Recognition with Speed-Up Techniques

2.1 Feature Vector-Based Japanese Character Recognition

Feature vector-based character recognition is widely used in Japanese OCR. In the sequence of text recognition, a text line is segmented and character candidate images are extracted at first. Some preprocessings (ex. size normalization) are applied to each character image, and a high-dimensional feature vector is extracted from the processed image. We use Peripheral Local Moment (P-LM) feature represented by 576-dimensional vectors as it was also used in [5].

In the proposed method, multiple fonts are used when making cluster data in order to take into account the distribution of feature vectors. Representative vector, which is the mean vector of the feature vectors of all fonts in each character, is used in the character recognition. Although there is some possibility to achieve higher accuracy by using a multi-font dictionary or another method to obtain representative vectors, using the mean vectors is quite efficient with respect to memory consumption and accurate enough at least for rough classification.

Let the kth representative vector in the dictionary be

$$\boldsymbol{m}^k = (m_1^k, m_2^k, \ldots, m_N^k)^{\mathrm{T}} \ , \tag{1}$$

where N is the dimensionality of the feature vector.

The similarity between the representative vector and the feature vector $\boldsymbol{v}$ of an unknown character image to be recognized can be evaluated by a distance measure such as Euclidean one. The class (character) of the top candidate can be found by

$$k_1 = \mathrm{argmin}_{1 \le k \le N_{class}} d(\boldsymbol{v}, \boldsymbol{m}^k) \ , \tag{2}$$

where $d(\ldots)$ represents a distance function, and N_{class} is the number of classes which is equal to the number of all characters N_{char}. The rth candidate k_r can be found by searching for the candidate with the rth shortest distance. In order to achieve faster processing, we may use squared Euclidean distance instead.

$$d(\boldsymbol{v}, \boldsymbol{m}^k) = \sum_{i=1}^{N} (v_i - m_i^k)^2 \ . \tag{3}$$

A certain way for evaluating (2) is to use the matching based on the full linear search (or exhaustive search) algorithm. Since Japanese/Chinese OCR engines need to handle more than 3,000 different characters, the linear search having $\mathrm{O}(N_{class} \cdot N)$ time complexity takes a long time.

2.2 Speed-Up Techniques

Several approaches have been proposed for improving the speed of general NN search problems.

Sequential Similarity Detection Algorithm (SSDA) [8] aims at reducing the computational cost in the distance calculation without introducing any error. SSDA keeps the past shortest distance during the linear search. The loop operation of the vector distance calculation for each class is aborted if the current accumulated value exceeds the shortest distance in the already-processed data. The distance comparison takes place at every s iterations, and we set $s = 8$ empirically in our work.

In ANN search, the data space is recursively partitioned into clusters and a search tree is constructed. The query data traverse the search tree from the root and, when it reaches the leaf node, the distances are evaluated only with the near data. There is a control parameter ϵ.

LSH scheme aims at reducing the number of the distance computation [4]. Some accuracy drop is observed in LSH due to its nature when it is applied to the character recognition task in which the intra-class variance is great.

Dimensionality reduction technique can also be used for improving the processing speed. It is known that the dimensionality of the raw feature vector is often too high for precise character recognition. The dimensionality could be made smaller to some extent by a subspace method. The smaller dimensionality contributes directly to higher processing speed. Note that various dimensionality reduction techniques can be combined with our proposed method.

3 Tree-Based Accuracy-Keeping Clustering and Fast Character Recognition

3.1 Overview

Based on Sobu's method [5], we have developed an improved version of the fast character recognition method consisting of the following three stages.

- 1st stage candidate reduction using the binary search tree constructed by an LDA-based accuracy-keeping clustering.
- 2nd stage candidate reduction using the conventional dimensionality reduction technique.
- Fine classification using the conventional linear search.

The final part may be replaced with another fine classifier for better accuracy. Note that the number of character candidates at the final stage is very small compared with the entire character set, and the total processing speed may not be affected so much even if a slow classifier is used here. The second stage has been changed from the original method [5] since the previous method was found being less accurate for handwritten character recognition. We have introduced the following new techniques [7].

- Best axis selection in the LDA-based clustering.
- Adaptive overlap optimized for each character class.

3.2 Accuracy-Keeping Clustering

We construct a hierarchical dictionary using LDA as shown in Fig.1.

Linear Discriminant Analysis. Multiclass LDA finds a small number of axes that can discriminate the classes well. Principal Component Analysis (PCA) is also well-known for the similar purpose. However, PCA may not find axes appropriate for efficient classification since the distribution of each class is not taken into account. The goal of LDA is to maximize the inter-class scatter matrix and minimize the intra-class scatter matrix. Therefore, linear discriminant axis is expected to separate the different classes effectively, and also to minimize the distribution of the same class. Sobu and Goto [5] reported that Canonical Discriminant Analysis (CDA), which is equivallent to the Multiclass LDA in this paper, outperforms PCA by 24.3% in speed, 1.0% in accuracy, and the CDA-based tree is much smaller (11.8%) than the PCA-based tree, using an electronically-rendered Japanese character dataset.

Throughout some preliminary experiments, we have found that sometimes the first axis is not suitable for separating the majority of characters properly, while the second or the third axis is more suitable. In order to achieve faster character recognition, we examine the second and the third axes as well, unlike Sobu's method[5], during the construction of the BST.

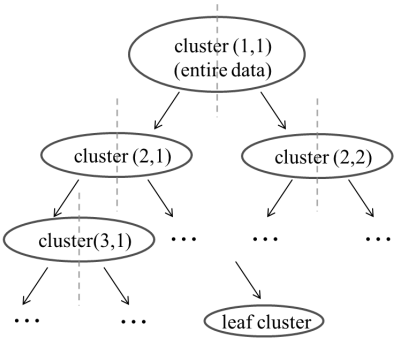

Fig. 1. Binary tree-based clustering

LDA-Based Search Tree. In LDA, the dimensionality of the feature vectors must be smaller than the number of classes. In order to allow small sub-classes during the clustering, we apply a dimensionality reduction before the LDA-based clustering. This can be achieved by using the PCA. When the dimensionality of the feature space is reduced to n by transform matrix $\boldsymbol{T}_n$ calculated by PCA, the kth representative vector in the dictionary is derived from (1) as

$$\boldsymbol{m}_r^k = \boldsymbol{T}_n \cdot \boldsymbol{m}^k = (m_1^k, m_2^k, \ldots, m_n^k)^{\mathrm{T}} \ . \tag{4}$$

We set the reduced dimensionality $n = 256$ to perform LDA precisely enough.

At the beginning, all the representative vectors of characters in the dictionary are put into the initial cluster (1,1) as shown in Fig.1. LDA is applied to the cluster and the major linear discriminant vector (axis) $\boldsymbol{v}_c$ is found. The projection of the character k onto the vector is given by

$$r_{k,c} = \boldsymbol{v}_c^{\mathrm{T}} \cdot \boldsymbol{m}_r^k \ , \tag{5}$$

which is a scalar value, where the suffix c is the cluster identifier. The cluster is cut into two pieces at the cut point

$$P_c = \frac{1}{N_{char}} \sum_{k=1}^{N_{char}} r_{k,c} \ , \tag{6}$$

which represents the hyperplane in the n-dimensional space. The linear discriminant vector is computed from each cluster, so suitable axis vector for each cluster can be obtained. The clusters are recursively cut as shown in Fig.1 until one of the following conditions are satisfied.

$$K_1 \geq N_c \ , \tag{7}$$

$$K_2 \leq \max(l_{c,1}, l_{c,2}) \ , \tag{8}$$

$$K_3 \leq (\text{depth of the node}) \ , \tag{9}$$

where N_c is the cluster size, K_1, K_2 and K_3 are pre-defined parameters, and $l_{c,i} = N_{c,i}/N_c$ $(i = 1, 2)$ is the child cluster size divided by the size of the current cluster assuming it is cut.

Here, we should take into account the second and the third axes as mentioned before. If the condition (8) is satisfied for the first axis, the second and the third axis are examined and the axis which can obtain lower $\max(l_{c,1}, l_{c,2})$ is chosen. The cluster cut stops if all the three axes satisfy (8).

In order to keep the accuracy, we allow some overlap between the clusters. The degree of the overlap was determined based on the variance of all the fonts and characters in our previous method [5]. In the proposed method, we take into account the variation of the intra-class vector variances.

Although the orientation of the obtained axis vector is expected to be suitable for many characters, the other characters may have wider distributions in the feature vectors along the obtained axis vector. Let $\boldsymbol{v}_r^k$ be the dimensionality-reduced feature vector of a font belonging to the character k. The interval $(\boldsymbol{v}_c^{\mathrm{T}} \cdot \boldsymbol{v}_r^k) \pm \alpha \cdot \sigma_k$ is calculated, where α is a pre-defined parameter and σ_k is the standard deviation within the same class k. If the cut point P_c falls in the interval, the character is registered to both clusters. If the different fonts of the same character fall in both clusters, the character is also registered to the both.

3.3 Dimensionality Reduction for Character Candidate Reduction

The number of candidates in each leaf node of the BST could be still large. For realizing the second stage candidate reduction, the representative vectors of all the characters are mapped into a low-dimensional space using PCA and kept as the low-dimensional dictionary data. The dimensionality of the new space, n_2, should be as small as possible to make the process faster. On the other hand, the accuracy of the rough classification deteriorates if n_2 is too small.

3.4 Fast Character Recognition Algorithm

In the dictionary, each node of the BST has a pair of $\boldsymbol{v}_c$ and P_c. Only the leaf nodes have the corresponding clusters, each of which is the set of character identification numbers.

Given an input character vector $\boldsymbol{v}$, the tree traversal begins at the root node. The inner product $\boldsymbol{v}_c^{\mathrm{T}} \cdot (\boldsymbol{T}_n \cdot \boldsymbol{v})$ is calculated and compared with the threshold P_c. When the traversal hits the bottom of the BST, the second stage candidate reduction is performed using the linear search with SSDA. Then K_4 number of Nearest Neighbor is chosen to perform linear search in the original high-dimensional space. The total time complexity is

$$\mathrm{O}(N \cdot n) + \mathrm{O}((\text{tree height}) \cdot n) + \mathrm{O}(N \cdot n_2) + \mathrm{O}(K_1 \cdot n_2) + \mathrm{O}(K_4 \cdot N) \ , \quad (10)$$

if the BST works efficiently. However, the fourth term can be much larger, $\mathrm{O}((\text{leaf node size}) \cdot n_2)$, if the separation of the classes is not good. Note that the times for matrix calculation in the dimensionality reduction, given by the first and the third terms, may not be so small compared with the other terms.

4 Experimental Results and Discussions

4.1 Setup

We used two hand-written character datasets, ETL9B for Japanese and HCL2000 for Chinese [6]. ETL9B dataset consists of 71 Hiragana, and 2,965 Kanji characters defined as JIS(Japanese Industrial Standards) Level 1. We picked up 2,965 Kanji character as experimental data. The number of data is 200, and we used the first 80 data for the dictionary, and the rest for the test data. HCL2000 consists of 3,755 characters. We used the first 80 data for the dictionary, and 120 in the next block for the test data. The PC used in the experiment is equipped with Athlon 64 3700+ (2.4GHz) and 2GB memory. To align the results with [5], we chose the parameter values $K_1 = 300$, $K_2 = 0.97$, $K_3 = 17$.

4.2 Performance Evaluation on ETL9B

Table.1 shows the accuracy and processing time for various n_2 and K_4 using ETL9B dataset. The accuracy is defined by the percentage of characters correctly recognized as the first candidates. The base accuracy and processing time were 94.35% and 16.92sec, respectively, using the full linear search.

When $n_2 = 40$ and $K_4 = 20$, the relative speed reaches 33.9 (times) at mere 0.24% accuracy drop, which is equivallent to 7.1 characters on average out of 2,965. The speed corresponds to 198.1 characters per video frame at 30fps. The average number of characters in the leaf clusters is 284, and it means that the tree-based classifier can effectively reduce the number of candidates by at most K_3 vector comparisons. The coverage of the first candidates in the BST, or the accuracy in other words, is 99.6%. About 41% of the total processing time is spent for the second candidate reduction. If we try another set of parameters, $n_2 = 50$, $K_4 = 40$, and $\alpha = 1.0$, the relative speed becomes 21.3.

Table 1. Accuracy(%) and Processing time(sec) for various n_2 and K_4 (ETL9B) (Sasaki2012 [7])

Accuracy (%) / Processing time(sec)				
		K_4		
		20	30	40
n_2	30	93.85 / 0.427	94.04 / 0.478	94.10 / 0.527
	40	94.11 / 0.499	94.17 / 0.550	94.20 / 0.598
	50	94.18 / 0.576	94.20 / 0.622	94.21 / 0.665

($\alpha = 0.5$)

Fig.2 shows the comparison result with various techniques. The parameter for LSH was chosen so the relative speed aligns with SSDA. Faster processing leads to worse accuracy. In the ANN search, the parameter was set $\epsilon = 3.5$. The proposed method is faster than Sobu's method (Tree) by 94.6%, and the accuracy is also much better.

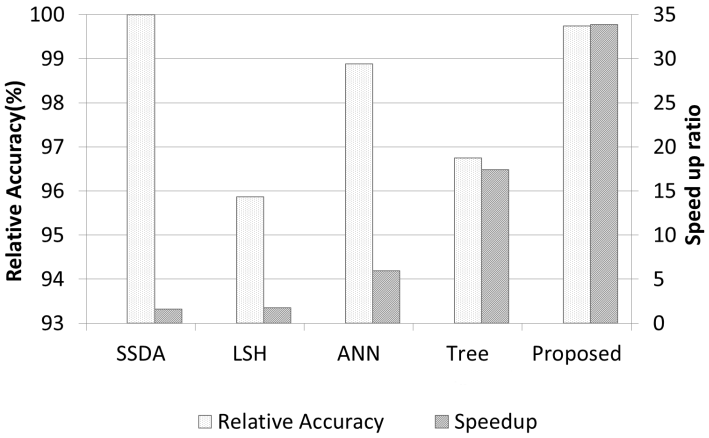

Fig. 2. Performance Comparison (ETL9B) (Sasaki2012 [7])

4.3 Performance Evaluation on HCL2000

We have evaluated the performance of the proposed method using HCL2000 dataset to investigate the possibility of extending our system to a multilingual one. The work is still under way and in an early stage.

Due to the limitation of our computer system, we had to set $\alpha = 1.0$. The other parameters are the same as those used in the experiments on ETL9B dataset. The base accuracy and processing time are 77.38% and 27.15sec, respectively, using the full linear search.

Table.2 shows the accuracy and processing time for various parameters. The low accuracy for the first candidates is probably due to the nature of HCL2000 data and to the fact that our system lacks a fine classifier suitable for Chinese. Table.3 shows the cumulative recognition rates up to the 10th candidates. We have obtained the data for up to $n_2 = 100$ and $K_4 = 100$ and found the 10th recognition rate saturates at 92.58%. This result suggests that the final accuracy could be improved by introducing a better classifier into the final classification stage.

Table 2. Accuracy(%) and Processing time(sec) for various n_2 and K_4 (HCL2000)

Accuracy (%) / Processing time(sec)				
		K_4		
		20	30	40
n_2	30	76.95 / 1.661	77.20 / 1.772	77.29 / 1.882
	40	77.26 / 2.231	77.33 / 2.328	77.35 / 2.418
	50	77.33 / 2.722	77.35 / 2.798	77.36 / 2.882

($\alpha = 1.0$)

Table 3. Cumulative recognition rates (%) (HCL2000)

		K_4		
rank		20	40	80
$n_2 = 40$	1	77.26	77.35	77.36
	2	84.39	84.60	84.63
	3	87.08	87.43	87.48
	4	88.53	89.02	89.09
	5	89.45	90.08	90.18
	6	90.08	90.85	90.98
	7	90.52	91.43	91.59
	8	90.84	91.89	92.08
	9	91.08	92.27	92.51
	10	91.08	92.27	92.51
$n_2 = 50$	1	77.33	77.36	77.36
	2	84.55	84.63	84.63
	3	87.33	87.47	87.49
	4	88.87	89.09	89.11
	5	89.87	90.17	90.20
	6	90.58	90.97	91.01
	7	91.08	91.57	91.63
	8	91.46	92.05	92.13
	9	91.76	92.46	92.56
	10	91.76	92.46	92.56

At $n_2 = 40$ and $K_4 = 20$, the relative speed is 12.2 (times), and this is much worse than ETL9B. The processing time corresponds to 56.1 characters per video frame at 30fps. A dissapointing thing is that the BST part does not contribute to the speed improvement so much for unknown reasons. The average number of characters in the leaf clusters is 2,583, which corresponds to 68.8% of the entire data. The coverage of the first candidates is 100%. It means that most of the speed improvement has come from the second candidate reduction stage. Further analysis is still ongoing for finding the main cause of the performance drop with HCL2000.

From another point of view, however, we could say our framework is safe with respect to undesired performance degradation in the tree-based classifier part.

5 Conclusion

Accurate and very fast multilingual character recognition is crucial for developing software for mobile devices, for realizing real-time OCR, and even for improving character segmentation performance. In this paper, we have introduced a tree-based clustering technique with LDA that can keep the character recognition accuracy quite high. The BST-based character recognition algorithm with candidate reduction in subspace runs at the speed 33.9 times faster than the full linear search with mere 0.24% accuracy drop on ETL9B, or 94.6% faster than

the conventional method. We have evaluated the performance using HCL2000 Chinese handwritten character dataset. A significant performance drop has been observed despite the similar shapes between Japanese and Chinese characters. Of course, we would like to spot the cause of the inefficient clustering with Chinese characters and to improve the proposed method further. A detailed analysis on typical error cases would help us improving the performance further, and it should be included in our future work.

A part of this work was supported by Grants-in-Aid for Scientific Research No.22300194 from JSPS.

References

1. Koga, M., Mine, R., Takahashi, T., Yamazaki, M., Yamaguchi, T.: Camera-based Kanji OCR for Mobile-phones Practical Issues. In: Proc. of ICDAR, pp. 635–639 (2005)
2. Mancas-Thilou, C., Ferreira, S., Demeyer, J., Minetti, C., Gosselin, B.: A multifunctional reading assistant for the visually impaired. EURASIP Journal on Image and Video Processing, 1–11 (2007)
3. Arya, S., Mount, D., Netanyahu, N., Silverman, R., Wu, A.: An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM 6(45), 891–923 (1998)
4. Datar, M., Immorlica, N.: P. Indyk, V.M.: Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In: Proc. of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262 (2004)
5. Sobu, Y., Goto, H., Aso, H.: Binary Tree-Based Accuracy-Keeping Clustering Using CDA for Very Fast Japanese Character Recognition. In: Proc. of MVA 2011, pp. 299–302 (2011)
6. Zhang, H., Guo, J., Chen, G., Li, C.: HCL2000 – A Large-scale Handwritten Chinese Character Database for Handwritten Character Recognition. In: Proc. of ICDAR, pp. 286–290 (2009)
7. Sasaki, T., Goto, H.: High-Accuracy Clustering Using LDA for Fast Japanese Character Recognition. IEICE Technical Report, PRMU2012–73, 19–24 (2012) (in Japanese)
8. Barnea, D., Silverman, H.: A Class of Algorithms for Fast Digital Image Registration. IEEE Trans. on Computers 2, C-21, 179–186 (1972)