Toward Font Identification for Printed Chinese Characters

Ming-Yu Tsai, Chao-Lin Liu, Kuang-Ting Chuang, Shang-Ming Huang
Department of Computer Science
National Chengchi University
Wen-Shan, Taipei 11605, Taiwan

Abstract

Authors of printed documents often strategically change fonts from one phrase to another to attract attention of the readers or signify special functions of the phrases. Typical optical character recognition (OCR) systems, however, remove font-related information from the scanned image while attempting to maximize the recognition accuracy and system performance. Consequently, we lose the font cues after we scan and recognize a printed Chinese document with such OCR software.

In fact, carrying out font identification does not have to place great extra burden on the OCR software. We can apply functions that are originally designed for character recognition to collect font-related information for font identification, thereby achieving both tasks at the same time. Specifically, we collect original and thinned images of 5401 most commonly used Chinese characters that are printed in five fonts and 12-point size, and build our system based on six font-related features extracted from these characters. We evaluate our implementation with documents, shown in Appendix B, that contain random and unformatted text printed in five fonts and three different sizes. Our system achieves a 97.3% average accuracy in these challenging tests.

1 Introduction

Fonts constitute an indispensable part of printed Chinese characters. Typical printed documents may employ a variety of fonts for different purposes. For instance, characters printed in **bold** or *italic* may attract more attention from readers of the documents. A sequence of words with a special meaning in its context may also be shown in a distinctive font, e.g., book titles in English technical papers are usually shown in italic. As a result, a good knowledge management system needs to consider the meaning conveyed by different fonts [16].

Despite these important communication functions carried by fonts, typical commercial optical character recognition (OCR) systems for Chinese characters do not preserve the original fonts in the scanned documents. All recognized characters would have been cast in one single typeface in the output documents. Such a loss in information can be undesirable for some applications, and manual post edition is in need to restore the original fonts.

Due to the potentially broad applications of OCR software, the literature has seen a correspondingly great amount of work in the field of printed and handwritten character recognition. For instance, Bunke and Wang put together a handbook on document image analysis [10], and Plamondon and Srihari survey hundreds of different approaches to handwriting recognition that are applicable to recognition of printed matters [13]. There are also more than one hundred domestic Master's and doctoral theses working on the recognition of Chinese characters, counting from the late 1980s, and a wide variety of approaches have been proposed. For instance, Lin takes advantage of the structural information from the scanned characters [4], Hsu applies the statistical properties, such as pixel density functions, collected from the scanned characters [6], and others attempt to integrate information of both kinds [5,15].

Most of the previous work, however, focus on recognizing the characters per se, and do not consider identifying the fonts. The thrust for this design decision is to reduce the search space for character candidates. Chinese characters can be printed in many different fonts, so research on character recognition typically attempt to ignore the font-related features for efficient character recognition. For instance, Lin encodes the structures of Chinese characters with *graph-associative memory* that would be built after the stage of thinning the original image of the characters [4]. Both Chuang and Yang consider the impact of different fonts on the recognition rate of OCR software, and implicitly encode the font-related information in the internal data structure of the recognition system [7,9]. Consequently, they do not concern with the problem of font identification.

The work on identifying fonts of printed matters can be traced back as far as 1989. Chang works on the recognition of printed English text, and his system can differentiate twenty fonts [8]. Fang applies the functions of pixel density for encoding font information. His system can tell which font is used for the forty extremely commonly used Chinese characters from Ming(細明體), Kai(楷書), Fong-song(仿宋體), **Black(粗黑體)**, and **Round (粗圓體)** [1].

In fact, identifying fonts can be an easy task for an OCR system. Many of the techniques designed for collecting structural and statistical information for character recognition can be applied to font identification. Identifying fonts does not have to dramatically increase the workload of an OCR system. We propose methods for differentiating Ming, Kai, Lee(隸書), **Black**, and **Round** for Chinese characters. Our methods take advantage of six distinctive features of these fonts in themselves. Some of the feature extraction operations rely on the existing thinning algorithm, so these features are not difficult to compute. Also, as we will discuss, these features are predictive in the sense that they work for Chinese characters that are unseen at the system training time. Our system achieves 97.3% accuracy for documents, shown in Appendix B, that contain characters randomly sampled from the 5401 most commonly used Chinese characters and printed in five fonts and three different sizes (12, 14, and 16 point words).

The following section briefly describes typical organization of an OCR system, and delineates the scope of our work in the whole system. Section 3 discusses the algorithm and font-related features that we use to identify fonts. We report experimental results of the algorithm in Section 4, and wrap up this paper with discussions in Section 5.

2 Problem Definition

Figure 1 shows the organization of a typical OCR system. Although the actual functions contained in the components may vary among different implementations, there are common functions in these components. At the preprocessing stage, we attempt to remove noise resulted from imperfect scanning or low quality document, extract individual characters from the image of the input document, normalize the sizes of characters to a standard size, and carry out the thinning operation on each character. Normalization is necessary for recognition techniques that employ information about pixel density and location functions. Thinning is a standard procedure for most OCR software, and one can find many references in the literature [e.g., 2,3,7]. Roughly speaking, the thinning procedure converts the images of characters to their skeletons, and that contributes a lot to the loss of font-related information. We design a simple character extraction algorithm that can remove a few simple types of noise in the scanned image. We also implement a normalization program that is good for our experiments, and we employ the thinning algorithm proposed by Lin [2]. We select this algorithm because it performs best in some harbinger tests.

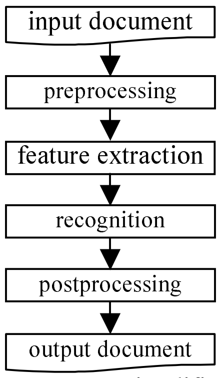

Figure 1: A simplified OCR system

We focus on the feature extraction and recognition stages in this paper, and we elaborate on this aspect shortly. Feature extraction is a very important component in an OCR system. Appropriate selection and successful extraction of the features from individual characters are essential for a high quality OCR system.

After extracting the features, the system compares the features with the internal encoding of characters in the database. Criteria are applied to determine which characters are "most similar" to the character being processed, and this basic principle applies to font identification too. The design of this comparison procedure varies from one system to another, and a very common strategy is to employ a hierarchical decision process at this step [e.g., 8,15].

Some OCR systems apply language models to select the character for hard-to-recognize characters in postprocessing. A common practice is to apply *N-gram* models to examine the context of the difficult characters, and select the candidate characters that best fit the context [12]. Although the contextual information can be useful for font identification, it is not the purpose of this paper to discuss this issue.

3 Feature Selection and Recognition

We present the features we propose for font identification. These features consider font-related features that appear in the original characters and their corresponding thinned characters. We call the characters that we obtain from applying thinning algorithms to the original characters *thinned characters*. As we discuss shortly, these features include thickness of original strokes, existence of staircases in thinned characters, terminal types of original strokes, corner types of original strokes, and shapes of bounding boxes of the original characters.

3.1 Feature Selection

We use the following enlarged sample characters to illustrate the font-related features that we use for font identification. These characters represent the same Chinese word in different fonts, i.e., from left to right, Ming, Lee, Kai, **Black**, and **Round**.

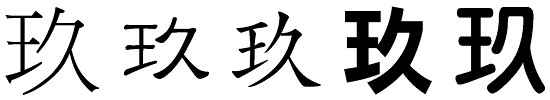

Figure 2: A Chinese character in five fonts

The first distinction we can observe is the varying thickness of the strokes in these fonts. As illustrated in the following chart, characters in **Black** and **Round** have much thicker strokes than others.

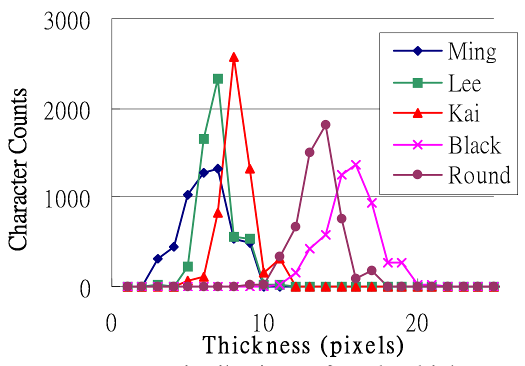

Figure 3: Distributions of stroke thickness

Applying the thinning algorithm proposed in [2] to the characters shown above will get corresponding thinned characters shown in the following Figure 4. Notice that the horizontal strokes of the character printed in Kai remain tilted. As a result, when we examine the horizontal strokes at the pixel level, the strokes would have been converted to the staircase shape, and this clearly can be used for font identification.

To detect and apply the staircase shape feature, we can examine the formation of the thinned horizontal strokes. The longest horizontal segment of

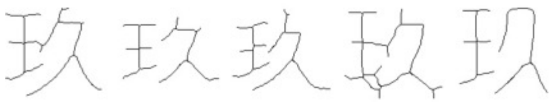

Figure 4: Thinning the characters in Figure 2

thinned characters in Kai is typically shorter than those of characters in other fonts. This is because short horizontal strokes in Kai will remain as short horizontal segments, but long horizontal strokes will be transformed into a group of short horizontal segments. Hence, the longest horizontal segment in the thinned character can be used as a feature for font identification. Moreover, if we compute the standard deviation of the lengths of the horizontal segments, thinned characters in Kai will have a smaller standard deviation than characters in other fonts. The following chart shows the distributions of standard deviations of segment lengths for different fonts. The curve for Kai clearly skews toward the left side.

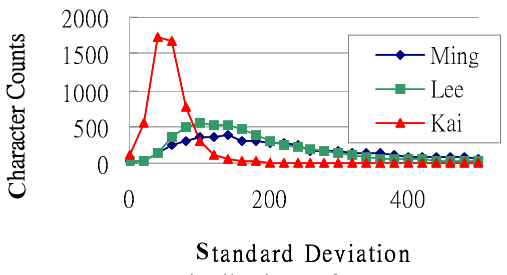

Figure 5: Distributions of segment lengths of different fonts vary.

The implication of the preceding observations is quite contradictory to one's instinct. Although thinning is originally designed for removing the influence of fonts on the appearances of characters, the results of thinning can be useful for font identification as well. The staircases created by our thinning procedure strongly indicate that the original characters are in Kai. In addition, we can compute the thickness of strokes with the contours of the original characters and the skeletons in the thinned characters. These functions make thinning a procedure useful for not only character recognition but also font identification.

Now take a closer look at the characters in **Black** and **Round** in Figure 2. Notice that we look for font-related features, so we are not interested in word-specific features, e.g., the intersecting locations of the strokes. As a result, we do not consider the different overall shapes of the right portions of these characters useful. The terminals of the strokes of these characters do shed light on the font classes. As its name has suggested, characters in **Round** have rounded terminals, while characters in **Black** do not demonstrate such a feature. Therefore, we can collect statistical information about the curvature of stroke terminals of characters printed in **Black** and **Round**, and apply this information for distinguishing these two fonts.

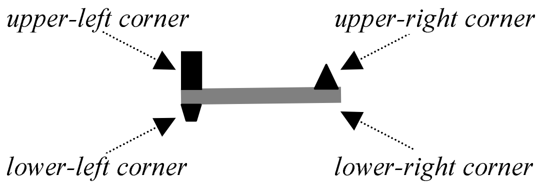

Figure 6: Encoding types of horizontal strokes

The shapes of the corners of horizontal strokes can be useful for font identification, and we classify the shapes into four categories. Figure 6 shows an enlarged stroke. We decompose a horizontal stroke into five components: the body, shown in gray, and four corners. The figure also illustrates different categories of the corners that may appear in a horizontal stroke. The upper-left, upper-right, lower-left, and lower-right corners respectively show a component that is taller than 9 pixels, between 4 and 8 pixels, between 1 and 3 pixels, and simply flat. These four types are assigned codes 2, 4, 3, 1, respectively. The type of a horizontal stroke is encoded by the codes of its four corners in the format (upper-left, upper-right, lower-left, lower-right), so the stroke shown in Figure 6 is of type (2,4,3,1). An examination of the characters in Figure 2 reveals that characters in Lee, **Black**, and **Round** are likely to have strokes of type (1,1,1,1). On the other hand, Ming and Kai are more likely to have special stroke types. We provide more details on how we apply this feature in Section 3.2.

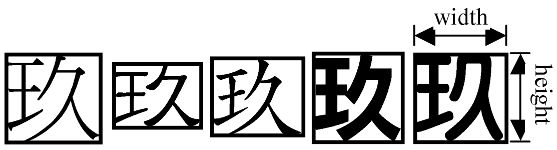

Figure 7: Shapes of bounding boxes differ.

Finally, examining the bounding boxes of characters shown in Figure 7 should persuade one to employ the height-to-width ratios of bounding boxes of characters for font identification. The observation is particularly self evident for characters in Lee that typically demonstrates low height-to-width ratios, and is confirmed by the statistics collected from actual data shown in Figure 8.

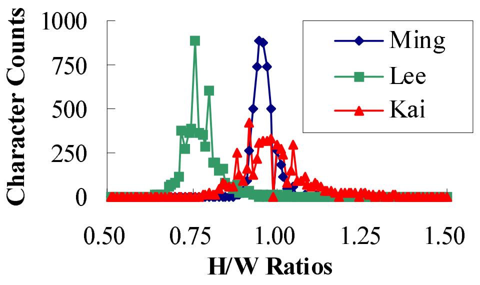

Figure 8: Distributions of height/width ratios

3.2 Recognition

We systematically extract the aforementioned features from the characters and classify their fonts. As shown in Figure 9, the top-level organization is a decision tree [17]. At the first step, a character whose average thickness of strokes is greater than a selected threshold will be classified into the **Black** and **Round** group. Otherwise, it will fall into the Ming, Lee, and Kai group. This threshold is chosen from the training data using statistical methods.

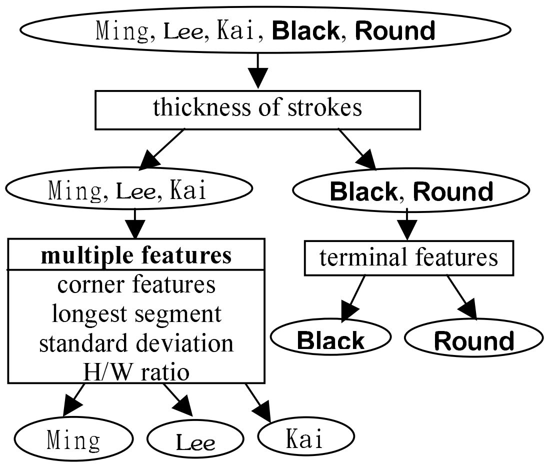

Figure 9: Skeleton of the identifier

If a character is classified into the **Black** and **Round** group, we use the curvature of terminals of the horizontal strokes to decide its font. Again, we choose a threshold for the curvature from the training data using statistical methods.

If a character is classified into the Ming, Lee, and Kai group, we utilize four different features to determine its font. We have tried several ways to integrate these features, and the best performing model so far is to apply decision rules [17] for font identification. For the corner features, we print and scan the training data in different fonts, record the types of horizontal strokes of each scanned characters, and compute the distributions of these three fonts for every stroke types. Each stroke type will be assigned the font that has the most number of strokes with this stroke type. Since each stroke has four corners and each corner may belong to one of four categories, there are 256 different stroke types, and each type is assigned a font. For the other three features, we select thresholds for them by statistical methods.

We list the decision rules in the decreasing priority in Appendix A. The rules are applied sequentially, and a font will be selected once a particular rule applies to the character being processed. Each rule has the precondition and result parts. The result part determines the font when the preconditions are all met. In the precondition, `corner→font` means that the majority of stroke types suggest that the character be in `font`. Also, we use `Max`, `standard`, and `ratio` to respectively denote the maximum segment length, standard deviation of the segment lengths, and height to width ratio that we define in the Section 3.1. The unit for `Max` is in number of pixels. Take the third rule for example. It classifies the character into Lee, if (1) the corner feature suggests that the character be Lee, (2) the height to width ratio is smaller than 0.89, and (3) the longest horizontal segment in the thinned character contains no less than 20 pixels.

4 Experimental Results

To test the effectiveness of our ideas, we set up an environment for experiments.[†] We selected the thresholds based on information collected from the images of the most commonly used 5401 Chinese characters. For this training purpose, we printed the characters in 12-point size and five fonts. These characters were then normalized to the size of 128 by 128 pixels, while preserving their height to width ratios. Depending on the shapes of the original characters, the normalization step would enlarge or shrink the original images until (1) the normalized image was contained in the 128 pixels by 128 pixels box, and (2) either the vertical or the horizontal direction of the normalized image just occupied 128 pixels. After this normalization step, we extracted features and computed the statistics from the normalized images. Finally, we subjectively chose the thresholds based on the statistics, hoping to improve the system performance when we applied the completed system to the 5401 characters printed in 12-point size and five fonts.

4.1 Experiments using Training Data

The following table presents the results. The leftmost column shows the actual font classes of the characters. Each number in a row means the number of characters that were classified into the font class in its corresponding column head. For instance, out of 5401 characters printed in Ming, our system classified 4747 characters correctly, but respectively misclassified 93 and 559 characters into Lee and Kai. After intensively tuning the parameters, our system achieved accuracy of 87.9%, 91.0%, 89.2%, 71.9%, and 68.6% for Ming, Lee, Kai, **Black**, and **Round**, respectively.

Table 1: Test results for 12-point sized characters

	Ming	Lee	Kai	**Black**	**Round**
Ming	**4747**	93	559	2	0
Lee	285	**4916**	199	1	0
Kai	278	304	**4818**	0	1
Black	2	2	0	**3886**	1511
Round	7	3	1	1684	**3706**

Besides the accuracy, there are few more points worth some analyses. First, the result matrix suggests that our method for differentiating the Ming, Lee, and Kai group and the **Black** and **Round** group is quite effective. Only 4 out of 16203 (i.e., 3*5401) characters in the Ming, Lee, and Kai group were classified incorrectly, and only 15 out of 10802 characters in the **Black** and **Round** group were classified incorrectly.

On the other hand, it is much harder to distinguish characters in the **Black** and **Round** group. Examining the raw data in more detail, we find that the relatively poor accuracy is a result of our using one hard-coded threshold of curvatures of terminals of horizontal strokes. This feature actually performs better for characters with less number of strokes. These characters have relatively thicker horizontal strokes, so there is a clearer difference in curvatures. When the characters become complex in terms of number of strokes, this distinction gradually disappears. Consequently, characters with more strokes contribute a lot to the inaccuracy for distinguishing characters in **Black** and **Round**.

The upper left corner of the result matrix shows a good reward to our using a complex set of decision rules based on only four features. The average accuracy for distinguishing Ming, Lee, and Kai is better than 89.4%.

[†] We printed both training and test data with an HP LaserJet 2100, and scan them with a Microtek ScannerMaker 4700 in 600 dpi mode.

4.2 Experiments using Test Data

To examine the flexibility of our approach, we tested our system with the 5401 characters in the five fonts and three different sizes, i.e., 12, 14, and 16 points. Recall that we tuned our system with 12-point characters, so the thresholds may not directly apply to characters of larger sizes. An intuitive solution to this challenge is to adjust thresholds that change with actual sizes of characters. This dynamic adjustment can be carried out by dividing the actual size of the image of the character being process with the average size of the 12-point characters, and adjusting the thresholds based on this ratio. The experimental results are shown in Tables 2 and 3.

Table 2: Test results for 14-point sized characters

	Ming	Lee	Kai	**Black**	**Round**
Ming	**4889**	56	455	1	0
Lee	244	**4998**	156	2	1
Kai	254	303	**4844**	0	0
Black	13	8	1	**3710**	1669
Round	5	3	1	1201	**4191**

Table 3: Test results for 16-point sized characters

	Ming	Lee	Kai	**Black**	**Round**
Ming	**5004**	31	364	2	0
Lee	243	**5034**	123	0	1
Kai	159	101	**5138**	0	3
Black	45	16	2	**3654**	1684
Round	15	10	2	1124	**4250**

Overall, our system adapted rather well to these new problems. Although the sizes had changed, the accuracy did not change significantly for most cases. The results in these tables support our previous observation that it is easier to distinguish between the group with thinner strokes (Ming, Lee, and Kai) and the group with thicker (**Black** and **Round**) strokes. Only around five characters in the thinner group were classified into the thicker group, and less than 100 characters were misclassified into the thinner group. Again, it is harder to distinguish between the fonts within the thicker group than those within the thinner group. For the experiments using 14-point and 16-point characters, we respectively achieved average 90.9% and 93.7% accuracy within the thinner group, but only 73.1% and 73.2% within the thicker group.

Finally we tested our system with documents containing characters in mixed fonts and sizes. Appendix B shows two pages of test data. Each page contains 150 characters that were randomly sampled from the 5401 characters in five fonts and three sizes. The accuracy for the first and the second page were both 97.3%. Considering that we only employed six features and tuned our system with 12-point characters, we consider this accuracy very encouraging.

5 Discussions

Our font identifier classifies fonts based on six font-related features. The thresholds for the features are trained based solely on 12-point characters. We tested our system with characters printed in five fonts and three sizes. The classifier achieves an average 83.7% accuracy in comprehensive open and close tests, and 97.3% for two randomly generated documents. In fact, in an experiment not reported in this paper, our system achieves comparable accuracy in identifying fonts of characters that do not belong to the most commonly used 5401 characters. Using font-related features allows our system to classify the fonts of unforeseen characters.

The current system can be enhanced in a few conceivable ways. Expanding our system to classify more fonts is the first alternative, and, as a result, we would need to add employ more features for the task. Moreover we would like to automate the design of font identifiers. As we discuss in Section 3.2, we manually tune the parameters for the rules listed in Appendix A. Although we seem to have tuned the system rather well, the tuning task is resource consuming, and does not necessarily lead to the best design. Covering algorithms, such as the PRISM algorithm discussed in [11,17], should provide a more efficient and systematic way to find a set of more effective rules.

We choose the decision tree shown in Figure 9 based on outcomes of a set of experiments. We actually implemented six decision trees that have different structures and thresholds, and the one shown in Figure 9 performs the best. Automatic construction and optimization of decision trees is not a new topic in the machine learning community. C4.5, a descendant of ID3, is a good example [14], and we plan to exploit this approach.

We believe that we can boost the classification accuracy and system efficiency by applying contextual information in font identification. Normal documents, such as newspaper and magazine articles, will follow ordinary styling principles, so they will not be as random as our test documents. Considering such style constraints in a font identifier will improve the efficiency.

We have begun to introduce these techniques from machine learning and natural language processing fields into our system, and we will report results of our experiments later when they are available.

Acknowledgements

The authors are grateful to Professor Wen-Hung Liao of the Department of Computer Science of National Chengchi University for his idea to work on this subject. This project was supported in part by the Grant NSC-89-2213-E-004-01 from the National Science Council of Taiwan.

References

[1] 方御凡，*中文字型識別、常用字辨認、與手寫字或印刷字之判別*，國立中央大學，電機工程研究所，碩士論文，1997。

[2] 林志勇，*二值影像之改良型細線化演算法*，中原大學，電子工程研究所，碩士論文，1999。

[3] 林昭興，*邊緣侵蝕法於數位影像之細線化*，中原大學，電子工程研究所，碩士論文，1997。

[4] 林基榮，*應用圖形關聯性對應方法辨識多字型印刷體中文字*，大同工學院（1999 年升格為大同大學），電機工程研究所，博士論文，1998。

[5] 高文忠，*手寫工整中文字辨識—統計法與結構分析法之整合研究*，國立台灣大學，電機工程研究所，博士論文，1996。

[6] 徐永煜，*貝氏決策型網路於手寫中文字辨識之研究*，國立交通大學，資訊工程研究所，碩士論文，1997。

[7] 莊富任，*應用掃描細線化演算法於中文文字辨識*，國立交通大學，資訊科學研究所，碩士論文，1999。

[8] 張鴻德，*應用個人電腦之多字型英文字數字辨識系統*，國立成功大學，電機工程研究所，碩士論文，1989。

[9] 楊和興，*具字型適應能力之印刷中文字辨識*，國立臺灣大學，電機工程學研究所，碩士論文，1997。

[10] H. Bunke and P. S. P. Wang (Editors). *Handbook of Character Recognition and Document Image Analysis*. World Scientific Publishing, 1997.

[11] J. Cendrowska. PRISM: An algorithm for inducing modular rules, *International Journal of Man-Machine Studies*, **27**(4), 349–318, 1987.

[12] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[13] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(1), 63–84, 2000.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[15] Y. Y. Tang, L.-T. Tu, J. Liu, S.-W. Lee, W.-W. Lin and I.-S. Shyu. Offline recognition of Chinese handwriting by multifeature and multilevel classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(5), 556–561, 1998.

[16] M. Travers. A visual representation for knowledge structures, *Proceedings of the Second Annual ACM conference on Hypertext*, 147–158, 1989.

[17] I. H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann, 2000.

Appendix A: decision rules for Ming**,** Lee**, and** Kai **listed in decreasing priority**

```
1. if (ratio<0.75) font=Lee
2. if ((corner→Lee)&&(ratio<0.83))
   font=Lee
3. if ((corner→Lee)&&(ratio<0.89)
       &&(Max>=20) font=Lee
4. if ((standard>120)
   &&(ratio<0.87)) font=Ming
5. if ((standard<120)&&(Max<=40))
   font=Kai
6. if ((corner→Ming)&&(Max>=50))
   font=Ming
7. if (corner→Lee) font=Lee
8. if ((corner→Kai)&&(Max<=25))
   font=Kai
9. if (corner→Ming) font=Ming
10.if (corner→Kai) font=Kai
11.if ((standard>120)
       &&(ratio<0.87)) font=Lee
12.if ((standard<120)
       &&(ratio<0.87)) font=Kai
13.if (Max<=20) font=Kai
14.font=Ming
```

Appendix B: two pages of test data (shrunk to fit into one page)