PERGAMON

Pattern Recognition 34 (2001) 15–35

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Optical recognition of handwritten Chinese characters by hierarchical radical matching method☆

An-Bang Wang[a], Kuo-Chin Fan[b,*]

[a]*Department of Banking and Insurance, Private Takming Junior College of Commerce, Taiwan, ROC*
[b]*Institute of Computer Science and Information Engineering, National Central University, Taiwan, ROC*

Received 8 October 1997; accepted 8 October 1999

Abstract

In this paper, a radical-based OCR system for the recognition of handwritten Chinese characters is proposed. In our approach, a recursive hierarchical scheme is developed to perform radical extraction first. Character features and radical features are then extracted for matching. Last, a hierarchical radical matching scheme is devised to identify the radicals embedded in an input Chinese character and recognize the input character accordingly. Experiments for radical extraction are conducted on 1856 characters. The successful rate of radical extraction is 92.5%. The average time for radical extraction is 0.65 second per character. Experiments for matching process are conducted on two sets: training set and testing set, each set includes 900 characters. The overall recognition rate in our experiments is 98.2 and 80.9% (for training set and testing set, respectively). The average recognition time of our hierarchical radical matching scheme is 0.274 s. There are totally 4716 radicals in 1800 characters. In average, one character consists of 2.62 radicals. Each character will match 7.28 radical templates in average. Thus, each radical will match 2.77 radical temples. The experimental results reveal that our proposed method is feasible, flexible, and effective.

Keywords: Radical extraction; Character pattern detection; Radical matching; Knowledge matching

1. Introduction

The recognition of optical handwritten Chinese character is an intricate problem. The reasons are due to the facts of large vocabulary set, high complexity, many mutually similar characters, and great variations for different handwriting styles. In this study, the intrinsic characteristics of off-line handwritten Chinese characters, such as the manner of writing and representing them, the location of radicals embedded in a Chinese character, etc., are investigated and used to solve this tough problem.

Since Chinese characters are composed of some basic structural components called radicals, it is an intuitive way to decompose Chinese characters into meaningful radicals before recognition. If we can successfully extract radicals embedded in Chinese characters, many important information can be obtained. Using these useful information, we can filter out unsuitable radical templates. The number of templates needs to be matched are thereby greatly reduced. Besides, the complexity of recognition will also be simplified because recognizing radicals is much easier than recognizing the whole character. There are only hundreds of radicals among thousands of Chinese characters. Moreover, it can thoroughly overcome the radical translation and scaling problems after normalization.

There are many researches undergoing on the study of radical extraction. In 1966, Casey and Nagy [1] employed a two-stage process, coarse classification and fine

☆This work was supported by National Science Council of Taiwan under grant NSC 87-2213-E-008-009.

*Corresponding author. Tel.: (03) 4227151-4453; fax: (03) 4222681.

0031-3203/00/$20.00
PII: S0031-3203(99)00207-1

classification, to identify the identity of each individual character. Ogawa et al. [2] and Wang et al. [3], used relaxation method to extract partial pattern, i.e., radical, from the spatial distribution of handwritten characters. Furthermore, Babaguchi et al. [4] classified radical combinations into seven categories and defined their individual ranges. However, since they used fixed ranges in the classification of segments, some ambiguities occur frequently. Tanaka [5] considered the background data to solve the radical extraction problem. The extraction of radicals from handprinted Kanji characters can be achieved by using some gap models to build the extraction process. Jeng et al. [6] used contour tracing and dynamic blank strip tracing methods to detect gaps between the radicals in a character. With these detected character gaps, the radicals can be easily extracted. Cheng and Hsu [7,8] presented two radical extraction methods. One is the stroke decision (SD) method and the other is the background thinning (BT) method. In the SD method, the strokes of a character must be extracted first, then using the information of long stroke and projection to divide the input character into left-right or top-down type. The BT method is to find a continuous cursive dividing path instead of a straight path as used in the stroke decision method. They also used a window extraction method to join two disconnected paths. Chang and Liu [9] also proposed a method to extract radicals embedded in printed Chinese characters. They used contour tracing algorithm to detect loop contour. The closed region is a radical which can be extracted by using polygonal copy technique. Lee [10] also proposed an on-line radical extraction method. He used rectangular region to represent each segment and then checked the overlapping condition of two sequential rectangular regions for radical extraction. Lin and Fan [9] classified Chinese characters into six types based on line segment order and projection methods. However, both of these two methods are only suitable for on-line handwritten characters. Liao and Huang [11] checked the validity of the relation whether the radical is a sub-character of the input character. The least-squares-error estimation and some properties of Chinese characters are utilized to check whether the relation is satisfied. Han et al. [9] proposed a stroke clustering method to determine the pattern of multi-font printed Chinese characters in conjunction with the extraction of embedded radicals.

The idea of using radicals for recognition has also been considered by Zhao [12] where the two-dimensional extended attribute grammar is used to describe a Chinese character. He utilizes attributes attached to the grammars describing Chinese characters to enhance the ability of noise and distortion tolerance in element level, and to describe the relations between distorted elements. In the work of Lu et al. [13], a hierarchical attributed graph (HAGR) is used to represent a Chinese character and a cost function mapping a candidate to a model graph is introduced. This approach can tolerate the variations of HAGR which reflect the instability or variability of handwritten Chinese characters resulting from different writing styles. Kuo and Mao [14] presented a multi-layer perceptron neural network for Chinese character recognition by making use of radical segmentation method based on the shape structure of Chinese characters. Lee and Chen [15] proposed a method based on the stroke order of Chinese characters to decompose a character into some sub-characters for effective character recognition. Cheng and Chen [16] analyzed the radical stroke pattern to determine the existence of salient radical stroke, and then to recognize radicals from salient radical strokes by a backward procedure through hypothesis and knowledge-using tests. Tseng and Lee [17] proposed a knowledge-based radical extraction method for handwritten Chinese characters to speed up the execution and increase the efficiency of character recognition by using some knowledge about radicals. Hsieh [18] refined the radicals in Da-Yi input method into 209 radicals suitable for radical extraction. These radicals are represented by second-order on-line models. Based on the models, a Viterbi algorithm is applied to extract the possible radicals from the input character. Then all extracted radicals are arranged as a connected graph, in which an edge links two radical nodes providing that the two radicals have no common strokes. By finding the maximum clique, the most possible radicals can be identified. Since the radicals in the Da-Yi input method with respect to the refined radicals are known, the extracted radicals are mapped with the Da-Yi radicals and then the character identity is found from the radical-character mapping table of the Da-Yi input system.

In this paper, a radical-based OCR system for recognizing handwritten Chinese characters is proposed. In our approach, a recursive hierarchical scheme is developed to perform radical extraction first. Character features and radical features are then extracted for matching. Last, a hierarchical radical matching scheme is devised to identify the radicals embedded in an input Chinese character and recognize the input character accordingly. Using this approach, the complexity of off-line handwritten Chinese character recognition and the templates and the size of radical database will be reduced tremendously.

The rest of this paper is organized as follows. Section 2 describes the overview of our proposed OCR system. Section 3 demonstrates the recursive hierarchical radical extraction method. Extraction of word and radical features will be addressed in Section 4. A hierarchical radical matching scheme is devised in Section 5 to identify the radicals embedded in an input Chinese character and recognize the input character accordingly. Section 6 illustrates the experimental results. Finally, concluding remarks are given in Section 7.

2. Overview of our proposed system

In this paper, a radical-based OCR system for handwritten Chinese characters is proposed. The systematic diagram of our proposed OCR system is shown in Fig. 1. It includes three modules. They are recursive hierarchical radical extraction, feature extraction, and hierarchical radical matching modules. A recursive hierarchical scheme is first proposed in module 1 to perform radical extraction. The proposed scheme, which is mainly based on some useful Chinese character structure information and knowledge about radicals, includes three layers. They are character pattern detection layer, straight cut line detection layer, and stroke clustering layer. Next, character features and radical features are extracted in module 2. Last, a hierarchical radical matching scheme, which contains three phases, is proposed in module 3 to identify the radicals embedded in an input Chinese character and recognize the input character accordingly. The three phases in radical matching module are radical matching phase, knowledge matching phase, and whole character matching phase.

In this paper, we assume that the preprocessing process, which includes smoothing, thinning, and stroke extracting, has been finished. The method to perform the preprocessing had been proposed in the work of Wang et al. [19]. Thus, the input of our proposed OCR system is the straight-line strokes represented by the vectors of coordinates.

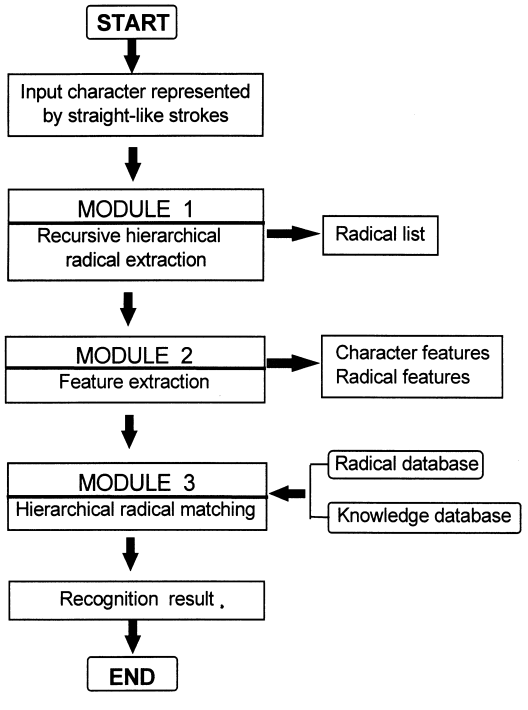

Fig. 1. System diagram of our proposed radical-based OCR system.

Experiments for radical extraction are conducted on 1856 characters. The successful rate of radical extraction is 92.5%. The average time for radical extraction is 0.65 s per character.

Experiments for matching process are conducted on two sets: training set and testing set, each set includes 900 characters. The overall recognition rate in our experiments is 98.2 and 80.9% (for training set and testing set, respectively). The average recognition time of our hierarchical radical matching scheme is 0.274 s. There are totally 4716 radicals in 1800 characters. On average, one character consists of 2.62 radicals. Each character will match 7.28 radical templates in average. Thus, each radical will match 2.77 radical temples. Experimental results reveal the feasibility of our proposed approach in recognizing handwritten Chinese characters.

3. Recursive hierarchical radical extraction module

There are four main concepts used in the radical extraction module. They are:

1. Character structure information: After stroke extraction, we will obtain the following character structure information:
   - for each stroke: length, orientation, location, number of cross points, and number of corner points.
   - relations between two strokes: crossed, connected, left, right, up, and down relations.
   - the relative location of each stroke in a character.
2. Some radicals have stable, salient structural features.
3. There exist gaps among radicals.
4. The cohesion property of strokes in a radical: strokes that belong to the same radical will be enclosed by a convex hull. Thus, radicals can be clustered in the center of the associating convex hull.

Based on these four concepts, a recursive hierarchical radical extraction scheme is developed. Fig. 2 illustrates the block diagram of the proposed scheme. It consists of three layers. Layer 1 is character pattern detection which is mainly based on concepts 1 and 2 to extract radicals embedded in Chinese characters with special pattern. Layer 2 is straight cut-line detection which is based on concept 3 to detect gaps among radicals. Using concept 4, a stroke clustering technique is devised in layer 3 to decompose Chinese characters that are left-right or up-down pattern into radicals. Some of these information will be implemented as rule-based knowledge in our system.

After layer 1, the radical embedded in Chinese characters with special pattern are extracted. The remaining component or the characters that are left-right or

up-down pattern will be processed in layers 2 and 3. The extracted radicals coupled with the remaining component will be built in a tree which is not necessarily a binary tree. And the extracted radicals (except generated by layer 1) together with the remaining component will be recursively fed back to the first layer to check whether it can be further decomposed. Finally, a radical list will be formed by the leaf nodes by tracing the generated tree using depth first search (DFS) technique.

3.1. *Layer 1*: *character pattern detection*

A Chinese character can be classified into one of the following 10 patterns as shown in Fig. 3: (1) *single-element* (SE), (2) *left-right* (LR), (3) *up-down* (UD), (4) *up-left* (UL), (5) *up-right* (UR), (6) *left-down* (LD), (7) *up-left-down* (ULD), (8) *left-up-right* (LUR), (9) *left-down-right* (ULD), and (10) *surrounding* (SU). Some examples illustrating the sample characters of these 10 patterns are shown in Fig. 4.

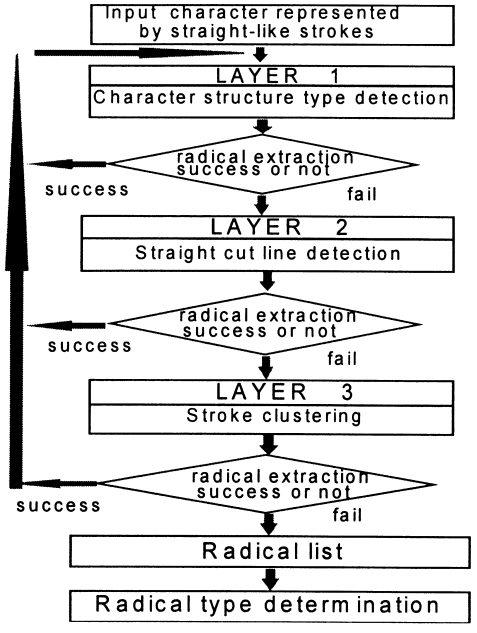

Fig. 2. Block diagram of the proposed recursive hierarchical radical extraction module.

The radicals of patterns 4–10 have stable and salient structural features. These structural features as mentioned in concept 1 are used in this layer to decompose the characters into two parts: the expected radical and the remaining component. Once the input Chinese character is classified as patterns 4–10, the expected radical will be extracted and identified. Hence, it does not need to be recognized in the matching phase.

In the following context, we will demonstrate the methods used in this layer in detail.

(1) *Pattern* 10 (*SU type*)*:* Chinese characters that belong to pattern 10 (such as 國, 因, 園) have salient radical located outermostly at the surrounding area. This outermost surrounding component is composed of one topmost horizontal stroke, one bottommost horizontal stroke, one left-most vertical stroke, and one right-most vertical stroke. And the lengths of these four strokes are long enough. A character is classified as a *SU type* character if we can detect the existence of this outermost surrounding component "┌┐". With the detection of a *SU type* character, it can thereby be decomposed into radical "┌┐" and the remaining part.

(2) *Pattern* 9 (*LDR type*)*:* Chinese characters that belong to pattern 9 (such as 函, 幽) have salient radical located outermostly at the left-down-right area. This outermost component is composed of one left-most vertical stroke, one bottommost horizontal stroke, and one right-most vertical stroke. And the lengths of these three strokes are long enough. A character is classified as a *LDR type* character if we can detect the existence of this outermost surrounding component "└┘". With the detection of a *LDR type* character, it can thereby be decomposed into radical "└┘" and the remaining part.

(3) *Pattern* 8 (*LUR type*)*:* Chinese characters that belong to pattern 8 have salient radical located outermostly at the left-up-right area. Therefore, the detection of this

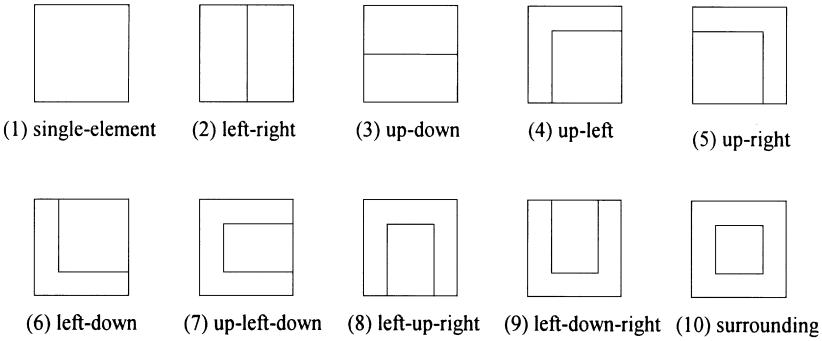

Fig. 3. Ten patterns of Chinese characters.

Type	Character	Type	Character
SE	中 車 更 生 身	LD	這 迎 道 建 趙
LR	的 個 時 地 清	ULD	區 匯 巨 匪 匿
UD	台 昂 雲 壁 篇	LUR	同 周 風 開 閉
UL	民 座 疾 房 虛	LDR	山 函 幽
UR	司 式 或 句 勾	SU	國 因 回 園 圖

Fig. 4. Examples illustrating each pattern of Chinese characters.

outermost component has to be performed first. Due to the variations of *LUR type* characters, it will be classified into four classes (types 81–84). For type 81 (such as 同, 周, 向, 岡), we have to detect the component "冂" first, and then decompose the characters into radicals "冂" or "𠂉" and the remaining part. For type 82 (such as 風, 鳳, 威), we detect the components "几" and "𠂆", and then decompose the characters into radicals "几" or "𠂉" and the remaining part. For type 83 (such as 開, 間, 鬧), we have to detect the components "門" and "鬥", and then decompose the characters into radicals "門" or "鬥" and the remaining part. For type 84 (such as 定, 受, 當, 勞), we have to detect the component "宀" first, and then decompose the characters into radicals "宀", "罒", "⺍", or "⺍" and the remaining part.

(4) *Pattern* 7 (*ULD type*): Chinese characters that belong to pattern 7 (such as 區, 匯, 臣, 匪) have salient radical located outermostly at the up-left-down area. This outermost component is composed of one topmost horizontal stroke, one bottommost horizontal stroke, and one left-most vertical stroke. And the lengths of these three strokes are long enough. A character is classified as a *ULD type* character if we can detect the existence of this outermost surrounding component "匚". With the detection of a *ULD type* character, it can thereby be decomposed into radical "匚" and the remaining part.

(5) *Pattern* 6 (*LD type*): Chinese characters that belong to pattern 6 have salient radical located outermostly at the left-down area. Therefore, the detection of this outermost component has to be performed first. Due to the variations of *LD type* characters, it will be classified into five classes (types 61–65). For type 61 (such as 這, 過), we have to detect the component "辶" first, and decompose the characters into radical "辶" and the remaining part. For type 62 (such as 建, 廷), we detect the component "廴", and decompose the characters into radicals "廴", and the remaining parts. For type 63 (such as 題, 匙), we have to detect the component "是", and then decompose the characters into radicals "是" and the remaining part. For type 64 (such as 起, 超, 趁), we detect the component "走" first, and then decompose the characters into radicals "走" and the remaining part. For type 65 (such as 勉, 甦, 旭, 魁), we have to detect the components "免", "更", "九", and "鬼" first, and then decompose the characters into radicals "免", "更", "九", or "鬼" and the remaining part.

(6) *Pattern* (*UR type*): Chinese characters that belong to pattern 5 have salient radical located outermostly at the left-down area. Therefore, the detection of this outermost component has to be performed first. Due to the variations of *UR type* characters, it will be classified into two classes(types 51 and 52). For type 51 (such as 可, 司, 句, 旬), we have to detect the component "丁", and then decompose the characters into radicals "丁", "冂", or "勹" and the remaining part. For type 52 (such as 戒, 式, 武, 裁, 載), we have to detect the component "弋", and then decompose the characters into radicals "戈", "弋", "弋", or "戋" and the remaining part.

(7) *Pattern* 4 (*UL type*): Chinese characters that belong to pattern 4 have salient radical located outermostly at the left-down area. Therefore, the detection of this outermost component has to be performed first. Due to the variations of *UL type* characters, it will be classified into two classes (types 41 and 42). For type 41 (such as 庫, 病, 慮, 曆), we have to detect the component "厂" first, and then decompose the characters into radicals "广", "疒", "卢", or "厂" and the remaining part. For type 42 (such as 居, 房, 眉), we detect the component "尸", and decompose the characters into radicals "尸", "戶", or "尸" and the remaining part.

An example illustrating the detection of pattern 10 characters is shown in Fig. 5. Examples illustrating the decomposition of pattern 8 characters are shown in Fig. 6.

Fig. 5. Detection of component "□" for characters with SU type.

Fig. 6. Characters with LUR type are classified into four classes: (a) detection of component "冂", (b) detection of components "几" and "𠂆", (c) detection of components "門" and "鬥", (d) detection of component "宀".

With stable and salient component being detected and its corresponding pattern being determined, some strokes that belong to the component but not be included must be grouped into the component. For example, after the component "戈" of the character "載" with pattern 5 has been detected, the strokes located above or intersected with the component must be grouped into the component to form the expected radical "𢦏".

In the character pattern detection process, some exception rules must be invoked to exclude the erroneous cases. These exception rules are heuristic and problem-dependent. Shown in Fig. 7 is an example illustrating this condition. Based on the knowledge about component "冂", the stroke s2 in Fig. 7(a) does not contain a 4-fork point, thus the three strokes s1, s2, and s3 can not be treated as component "冂" in this layer. Figs. 7(b) and (c) are the correct radicals extracted by using the technique to be discussed in next layer.

3.2. Layer 2: straight cut-line detection

Since gaps are existed among radicals embedded in a Chinese character, we can utilize these background information to achieve the radical separation and extraction goal. Since some gaps may be wide and clear, the middle cut line of this kind of gap will decompose the input character into two parts and will be called *clear straight cut line*. Whereas some gaps may be unclear but they are long enough for radical separation, we will treat this kind of gap as an one-pixel gap and call its cut line as *nearly straight cut line*. The detail description of straight cut line detection algorithm is given as follows.

Step 1: The character or sub-component of the character will be bounded first by a square.

Step 2: From the perimeter of the square, trace inwardly the input image after stroke extraction until a pixel is met in some stroke. Then count the tracing depth and record it.

Step 3: If we can trace through from one side to the corresponding side, then there exists a clear gap (*a clear straight cut line*). Measure the range of the gap. The wider the gap range, the more stable the gap will be. Otherwise, if the tracing depth is long enough (say, greater than 0.9 multiple height or width with respect to vertical or horizontal trace), then treat it as an one-pixel gap (*a nearly straight cut line*).

Step 4: After horizontal and vertical gaps have been detected, we must determine the combinative type of gaps. We classify the combinations into four types. Type 0 means that gap doesn't exist. Type 1 means that horizontal gaps exist. Type 2 means that vertical gaps exist. Type 3 means that both horizontal and vertical gaps exist.

Step 5: If the gap type is 0, then go to the next layer. If the gap type is 1, then the input image will be decomposed horizontally into some areas. If the gap type is 2, then the input image will be decomposed vertically into some areas. If the gap type is 3, then the input image will be decomposed horizontally or vertically into some areas according to certain rules. If it fails to be decomposed, then decompose it in the other direction.

Step 6: The areas containing one stroke will be merged to the nearest area according to the distance between these two areas. The areas containing two or three strokes will be checked in advance according to some knowledge about a radical to see whether it must be merged to the nearest area or remain unchanged.

Step 7: The resultant areas may contain a radical or just a sub-component. Hence, we must go back to layer 1 recursively to see whether it needs to be decomposed further or not.

An example illustrating the above procedure is shown in Fig. 8. After tracing inwardly from the perimeter to the

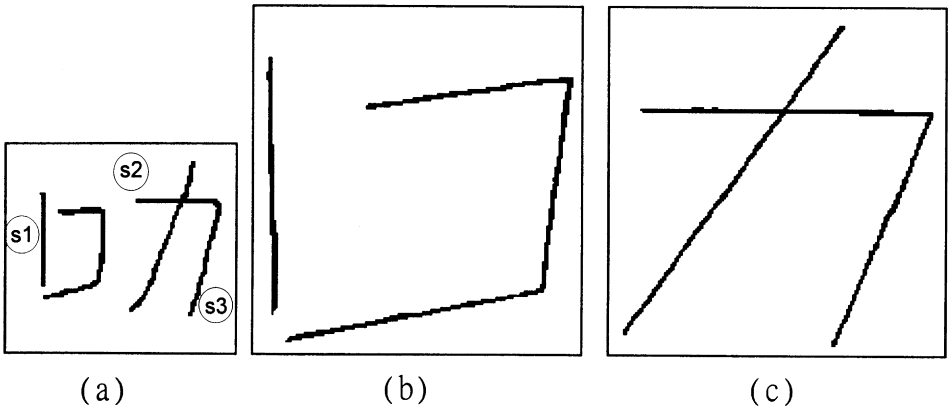

Fig. 7. Example illustrating how the exception rules are used to exclude the incorrect radical component.

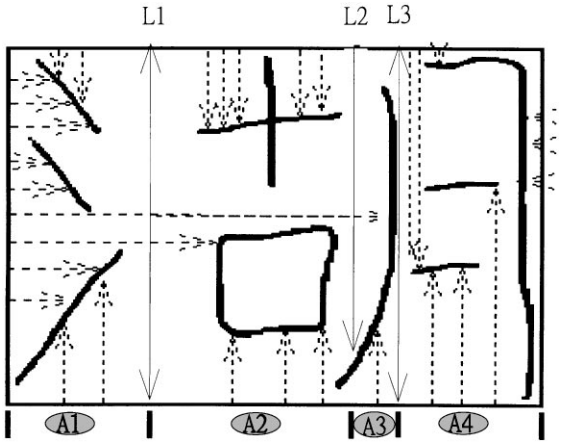

Fig. 8. Example illustrating radical extraction by making use of straight cut line. Lines with two arrows (such as L1 and L3) are clear straight cut lines. Line with one arrow (such as L2) is nearly straight cut line. The character "湖" are decomposed into four areas by these three vertical cut lines.

skeleton of the character "湖", we can find three vertical cuts lines: two clear straight lines (L_1 and L_3) and one nearly straight line (L_2). Thus, the character will be decomposed into 4 areas: A_1, A_2, A_3, and A_4. Area A_1 contains three strokes, area A_2 contains six strokes, area A_3 contains one stroke, and area A_4 contains four strokes. Area A_3 will be merged into area A_4 because the stroke in area A_3 is more closer to area A_4 than area A_2. Finally, we will obtain three areas A_1, A_2, $A_3 + A_4$. These three areas will be fed back to layer 1 to continue the decomposition process until all areas can not be decomposed any further. Four radicals ("氵", "十", "口", and "月") will thereby be obtained.

Shown in Fig. 9 is another example illustrating the case where both horizontal and vertical gaps exist. After tracing inwardly from the perimeter to the skeleton of the character "障" as shown in Fig. 9(a), we find one vertical cut line (L_1 as shown in Fig. 9(a)) and four horizontal cut lines (L_1, L_2, L_3, and L_4 as shown in Fig. 9(b)). As the vertical gap is wider than the other four horizontal gaps. Hence, the character will be decomposed vertically into two areas A_1 and A_2 containing "阝" and "章", respectively. The second component "章" as shown in Fig. 9(b) will be decomposed into five areas: A_1, A_2, A_3, A_4, and A_5. Areas A_1, A_2, and A_3 will be merged to form radical "立". Areas A_4, and A_5 will be left unchanged which represent radical "日" and "十", respectively.

3.3. *Layer 3*: *stroke clustering*

After layers 1 and 2, the character or sub-component must be SE, LR, or UD pattern. The results after layers 1 and 2 are then fed to the stroke clustering layer to

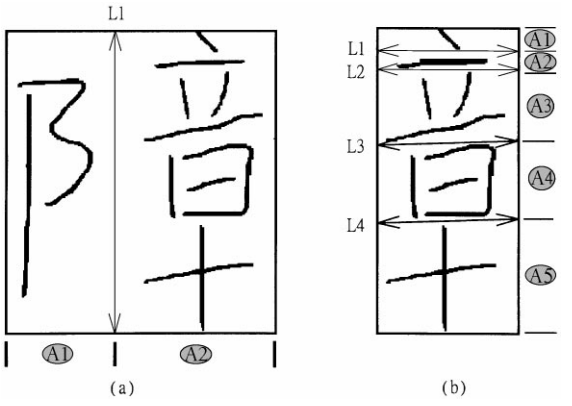

Fig. 9. Example illustrating radical extraction when both horizontal and vertical gaps exist.

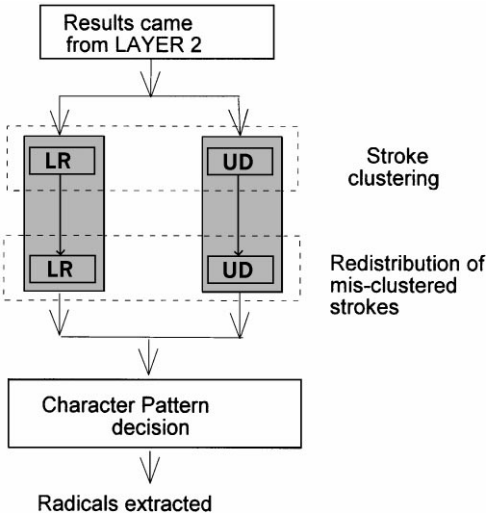

Fig. 10. Block diagram of stroke clustering layer.

extract radicals that belong to LR or UD pattern. The block diagram of stroke clustering layer is shown in Fig. 10. The stroke clustering technique adopted here was proposed by Han et al. [20] for extracting radicals of multi-font printed Chinese characters. We modify it to fit the requirement of our proposed method for handwritten Chinese characters. There are two modifications: (1) The input in the original method must be a whole character, whereas it can be a whole character or its sub-component in our modified method. (2) The process in the original method only performs once, but performs recursively in our modified method. For simplicity, the input in this subsection will be called input component which may be a whole character or a sub-component of a character.

Based on the relationship among strokes, the input character or sub-component is arbitrarily assigned as

a left-right (LR) or up-down (UD) character first. Each stroke of the input character is considered as a sample point, and then apply the K-mean stroke clustering algorithm to classify the input component into two clusters which represent the two radicals.

The clustering results may be erroneous. Thus, a mis-clustered stroke modification procedure is devised to rearrange the mis-clustered strokes. After the stroke clustering and mis-clustered stroke modification procedures, the dividing path of the input character for each previously assumed character pattern is obtained. Finally, an evaluation function is designed in character pattern decision procedure to determine the identity of the character pattern. Besides, the extracted radicals split via the dividing path of the identified character pattern are also obtained. The detail description of each procedure in stroke clustering layer is discussed in the following subsections.

3.3.1. Stroke clustering

Before performing K-mean clustering algorithm, two tasks have to be performed in advance. First, strokes connected with 4-fork points must be grouped to form stroke blocks to avoid ambiguous and erroneous clustering result. Each of the remaining strokes in the input component is also represented by a block. Second, the two dimensional coordinates (x_i, y_i) of the center points of all stroke blocks in the input component, where the origin is located at the upper-left corner of the input image, are next reduced to the samples with one dimensional data (t_i) by making use of the following equations.

(a) for left-right characters or components:

$$t_i = x_i, \quad i = 1, 2, \ldots, n, \tag{1}$$

(b) for up-down characters or components:

$$t_i = y_i, \quad i = 1, 2, \ldots, m, \tag{2}$$

where n and m are the numbers of stroke blocks.

Then K-mean clustering algorithm [21] is applied to cluster the reduced data into two clusters by assigning $K = 2$. An example shown in Fig. 11 is used to illustrate the stroke clustering procedure. Figs. 11(a) and (c) illustrate the stroke blocks of two characters "址" and "貨" which belong to LR and UD pattern, respectively. The black circle represents the center of each stroke block and the rectangle drawn with dashed lines depicts the range of each stroke block. The reduced data t_i for the characters in Figs. 11(a) and (d) are shown in Figs. 11(b) and (e) by utilizing Eqs. (1) and (2), which are the input samples for K-mean clustering procedure. The clustering results of stroke blocks as shown in Figs. 11(c) and (f) form two clusters (radicals) for left-right and up-down characters.

3.3.2. Redistribution of mis-clustered stroke

Although K-mean clustering algorithm minimizes the sum of square distances from all samples in a cluster to the cluster center, the clustering results may still be erroneous. The concept of redistributing the mis-clustered strokes generated by K-mean clustering algorithm is to detect an ambiguous area which is the overlapping area

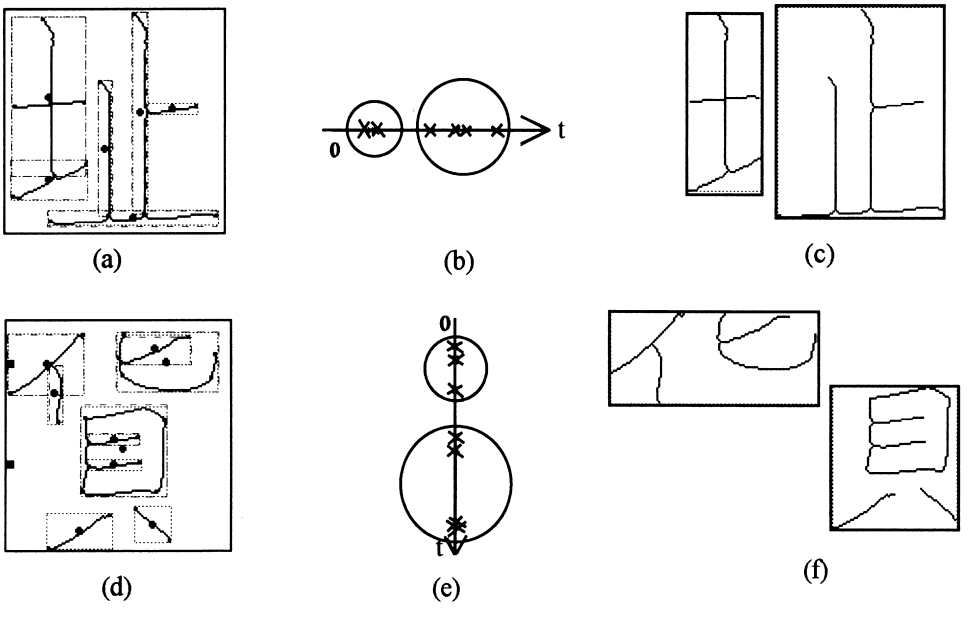

Fig. 11. Illustration of stroke clustering for characters with LR and UD patterns.

of two radicals (left-right or up-down) and then redistribute the stroke blocks in the ambiguous area to the correct cluster.

3.3.2.1. Redistribution strategies for left-right characters or components. Before stating the rearranging strategies for left-right characters, the ambiguous area, where the mis-clustered strokes frequently fall in, has to be defined first. Consider the two extracted radicals C_l and C_r for input component with LR pattern. Assume left radical C_l and right radical C_r contain B_i and B_j stroke blocks, respectively, where i is from 1 to m and j is from 1 to n. Each stroke block B_i can be represented by its upper-left corner $(x_1^{B_i}, y_1^{B_i})$ and lower-right corner $(x_2^{B_i}, y_2^{B_i})$ as denoted by $\{(x_1^{B_i}, y_1^{B_i}), (x_2^{B_i}, y_2^{B_i})\}$. Note that the subscripts represent the coordinates of the upper-left and lower-right corners, and the superscripts represent the corresponding stroke blocks. The ambiguous area which is the overlapping area of the left and right radicals can be defined as follows:

$$A_a = \{(x, y) | \min_{B_j \in C_r} (x_1^{B_j}) < x < \max_{B_i \in C_l} (x_2^{B_i}), 0 < y < height\}, \tag{3}$$

where value *height* denotes the height of the input character. If a stroke block falls in the ambiguous area over 40% in the x-direction for left-right characters or components, it is considered as an ambiguous stroke block and needs to be redistributed. The ambiguous stroke blocks B_k is assigned to cluster C_l or C_r by making use of the following criteria:

$$B_k \text{ is assigned to } \begin{cases} C_l & \text{if } d_l < d_r, \\ C_r & \text{if } d_l > d_r, \end{cases} \tag{4}$$

where

$$d_l = \frac{x_1^{B_k} + x_2^{B_k}}{2} - \max_{\{B_i \in C_l, B_i \notin A_a\}} (x_2^{B_i}) \quad \text{and}$$

$$d_r = \min_{\{B_j \in C_r, B_j \notin A_a\}} (x_1^{B_j}) - \frac{x_1^{B_k} + x_2^{B_k}}{2}.$$

Note that since the case for "$d_l = d_r$" seldom occurs in our experiment, we will arbitrarily assign the stroke block to its left side when it occurs. The redistribution of mis-clustered stroke process will result in the new left C_l and right C_r radicals. These two new radicals will be utlized to generate the vertical dividing path for left-right characters.

The vertical dividing path for left-right characters is thus defined as

$$X_v = \left(\min_{B_j \in C_r} (x_1^{B_j}) + \max_{B_i \in C_l} (x_2^{B_i}) \right) / 2. \tag{5}$$

3.3.2.2. Redistribution strategies for up-down characters or components. The redistributing rules for up-down characters are similar to those for left-right characters. Consider the two extracted radicals C_u and C_d for input component with UD pattern. Assume up radical C_u and down radical C_d contain B_i and B_j stroke blocks, respectively, where i is from 1 to m and j is from 1 to n. Each stroke block B_i can be represented by its upper-left corner $(x_1^{B_i}, y_1^{B_i})$ and lower-right corner $(x_2^{B_i}, y_2^{B_i})$ as denoted by $\{(x_1^{B_i}, y_1^{B_i}), (x_2^{B_i}, y_2^{B_i})\}$. Note that the subscripts represent the coordinates of the upper-left and lower-right corners, and the superscripts represent the corresponding stroke blocks. The ambiguous area which is the overlapping area of the up and down radicals can be defined as follows:

$$A_a = \{(x, y) | \min_{B_j \in C_d} (y_1^{B_j}) < y < \max_{B_i \in C_u} (y_2^{B_i}), \quad 0 < x < width\}, \tag{6}$$

The ambiguous stroke block B_k falling in the ambiguous area over 40% in y-direction has to be redistributed and assigned via the following criteria;

$$B_k \text{ is assigned to } \begin{cases} C_u & \text{if } d_u < d_d, \\ C_d & \text{if } d_u > d_d, \end{cases} \tag{7}$$

where

$$d_u = \frac{y_1^{B_k} + y_2^{B_k}}{2} - \max_{\{B_i \in C_u, B_j \notin A_a\}} (y_2^{B_i}) \quad \text{and}$$

$$d_a = \min_{\{B_j \in C_d, B_j \notin A_a\}} (y_1^{B_j}) - \frac{y_1^{B_k} + y_2^{B_k}}{2}.$$

The redistribution of mis-clustered stroke process will result in the new up C'_u and down C'_d radicals. These two new radicals will be utlized to generate the horizontal dividing path for up-down characters.

The horizontal dividing path Y_h for the new up C'_u and down C'_d radicals can be obtained accordingly by the following formula,

$$Y_h = \left(\min_{B_j \in C_d} (y_1^{B_j}) + \max_{B_i \in C_u} (y_2^{B_i}) \right) / 2. \tag{8}$$

Shown in Fig. 12 is an example illustrating how to redistribute the mis-clustered strokes. Fig. 12(a) illustrates the result generated by K-mean stroke clustering algorithm. There are two extracted radicals C_l and C_r, where C_l contains 5 stroke blocks (B_1, B_2, B_3, B_4, B_5) and C_r contains 2 blocks (B_6, B_7). The meshed area in Fig. 12(b) shows the ambiguous area which consists of 3 stroke blocks (B_3, B_4, B_5). Fig. 12(c) is the correct result after mis-clustered stroke redistribution procedure.

3.3.3. Character pattern decision

The dividing path which splits the input character or component into two clusters, called clusters A and B, are obtained from the methodologies as described in the previous section. The evaluation criteria to compute the

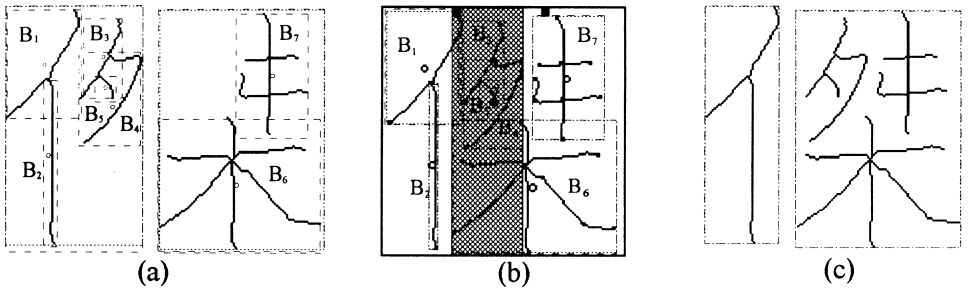

Fig. 12. Redistributing results for left-right characters.

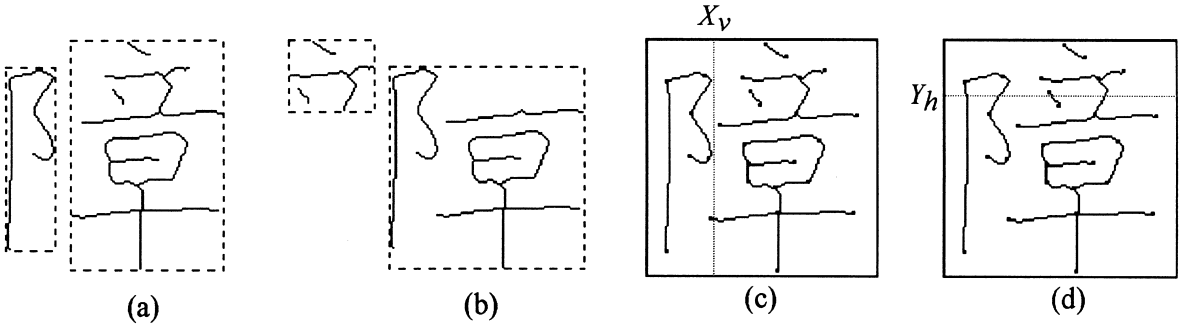

Fig. 13. Illustration of distance computation for character with LR pattern.

distance generated via the dividing paths are described as follows: Traverse the dividing path from the boundary of the input character. If the dividing path and the stroke S of cluster A meet at point P, traverse the partial stroke falling in the area of cluster B to pass the meeting point P, and then continue the unfinished traversing on the dividing path. On the contrary, if the dividing path meets stroke S which belongs to cluster B at point P, traverse the partial stroke falling in the area of cluster A to pass the meeting point P. The total traversed pixels N'_A belonging to the strokes of cluster A but falling in the area of cluster B over the total pixels N_A of cluster A is called the *ratio* of *mis-stroke error* and denoted as N'_A/N_A. On the other hand, the ratio of mis-stroke error for cluster B is denoted as N'_B/N_B. The total amount of mis-stroke errors $(N'_A/N_A) + (N'_B/N_B)$, called *distance*, is calculated to decide which type the input character is by choosing the type with the smallest distance. Furthermore, if the distances for left-right and up-down characters or components are both larger than a pre-selected threshold, the input character or component is assigned to the single-element pattern. In our experiments, the pre-selected threshold is arbitrarily chosen as 35 by experience.

To illustrate the procedure, a Chinese character that belongs to left-right pattern is given to demonstrate the computation of distances for left-right and up-down patterns. The left and right radicals generated via left-right radical extraction layer, and the up and down radicals generated via up-down radical extraction layer are displayed in Figs. 13(a) and (b), respectively. The dividing paths for left-right and up-down patterns are drawn using dotted lines as shown in Figs. 13(c) and (d). Based on the principles of distance computation, the distance of left-right pattern which equals 2 is smaller than the distance of up-down pattern which is 33. Therefore, we can declare that the identity of the input character is left-right pattern instead of up-down pattern.

3.4. Determination of radical type

According to the appearing location of a radical embedded in a Chinese character, a radical may be classified into one of the following 19 types as shown in Fig. 14. After radical extraction, we have to determine which type the extracted radical is. There are two purposes for this procedure. First, we can know the appearing location of a radical. Second, it can be used to distinguish two or more similar characters which consist of the same radicals but with different locations. For example, characters "陪" and "部", which consist of three same radicals "阝", "立", and "口", can be distinguished correctly according to this criterion.

The radical type of the radicals embedded in characters from pattern 4 through 10 will be treated as type 0.

An example is shown in Fig. 15 to illustrate the whole process of our proposed recursive hierarchical radical extraction scheme. Fig. 15(a) is the input character "歐". Fig. 15(b) illustrates the process of radical extraction and

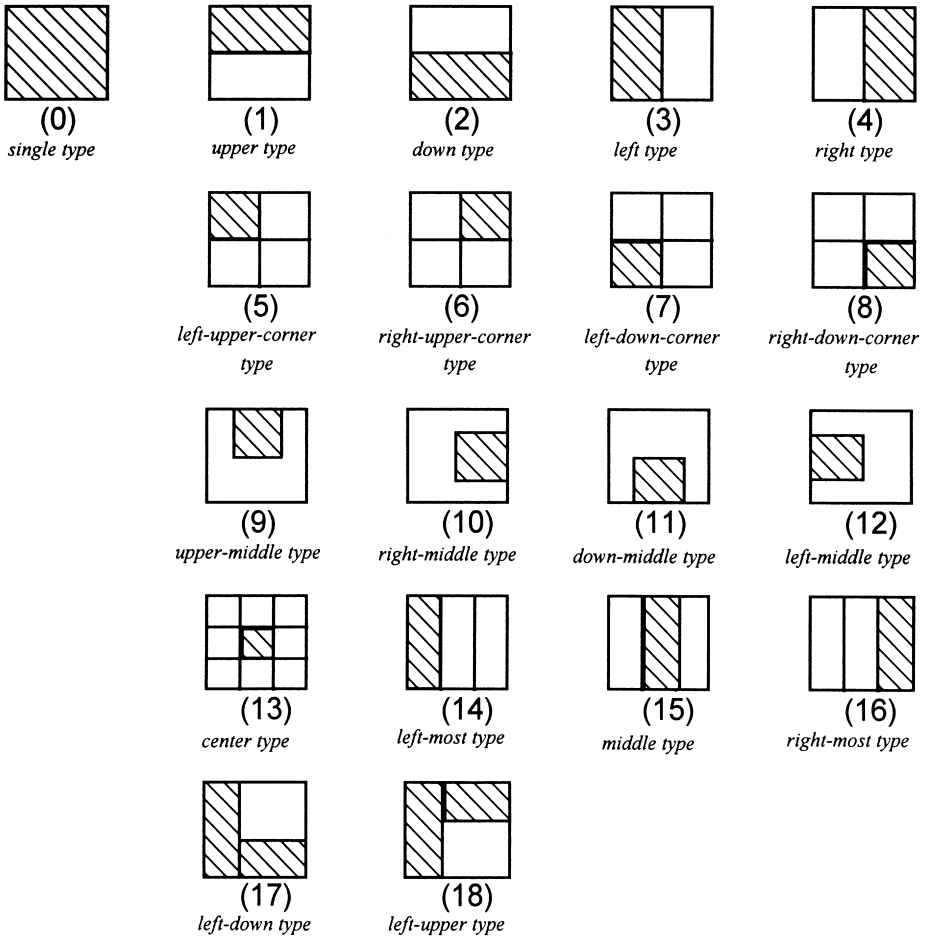

Fig. 14. The 19 appearing types of a radical.

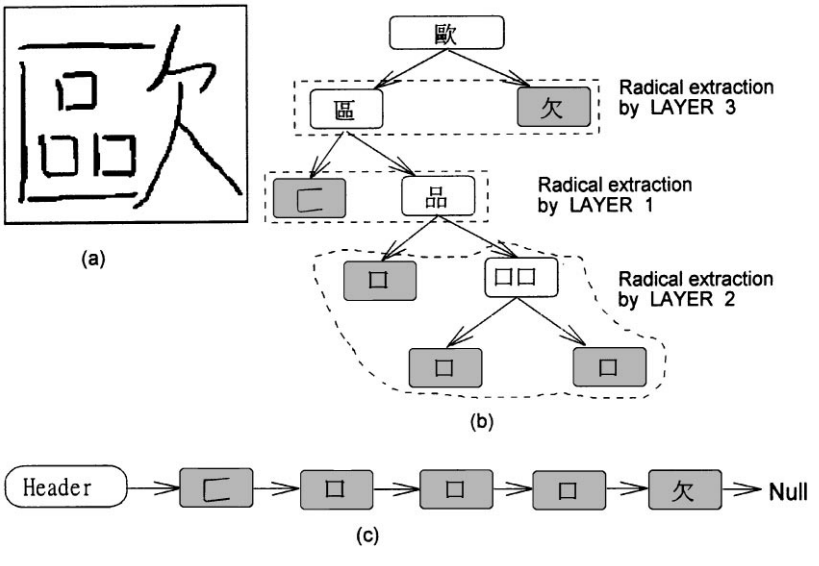

Fig. 15. (a) The input character, (b) the process of radical extraction from top to bottom where leaf nodes represent the extracted radicals, (c) the link list formed by the extracted radicals.

the built-up tree. Shown in Fig. 15(c) is a radical list formed by the leaf nodes when tracing the tree in depth first search (DFS) way.

Some examples are given in Fig. 16 to illustrate the results of radical extraction. Figs. 16(a)–(f) are characters with UL, UR, LD, LUR, LUR, and SU types, respectively.

4. Feature extraction module

To overcome the stroke variation problem of handwritten Chinese characters, characters after stroke extraction and radicals after radical extraction have to be normalized to 150×150 resolution before performing the matching process.

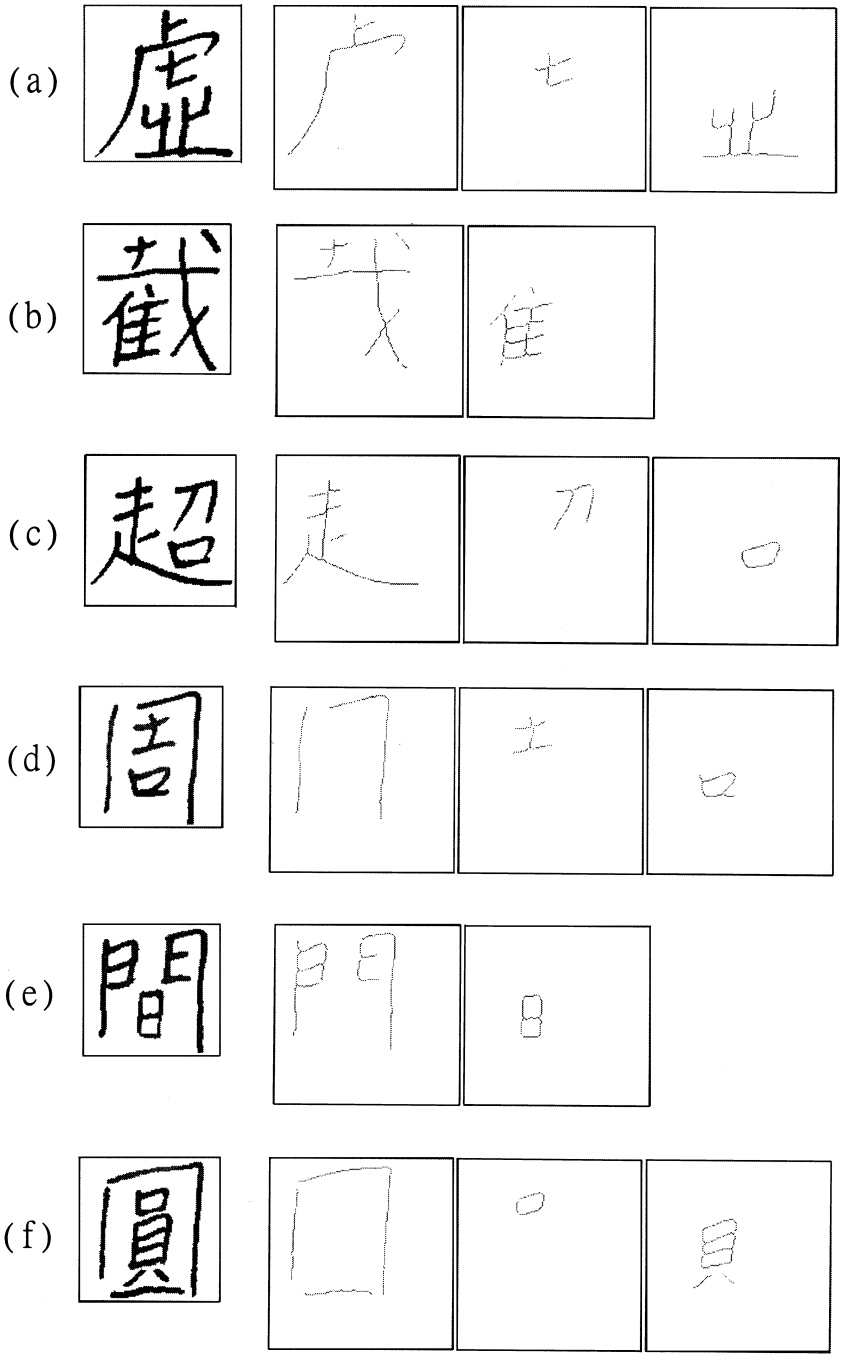

Fig. 16. Examples illustrating radical extraction result: (a)–(f) are characters with UL, UR, LD, LUR, LUR, and SU types, respectively.

There are two kinds of feature representation for each character, one representation is based on the strokes constituting the character and another representation is based on the radicals constituting the character. These two kinds of feature will be used in the whole character matching process and the radical matching process, respectively. The features extracted for each stroke constituting a radical or a character include stroke length, slope angle, the coordinate of center point, and the angle between any two strokes.

5. Hierarchical radical matching module

After radical and feature extraction, a hierarchical relaxation-based radical matching scheme is developed to perform radical identification and character recognition on scanned input Chinese characters. Shown in Fig. 17 is the block diagram of the proposed matching scheme which includes three matching phases. They are radical matching phase, knowledge database matching phase, and whole character matching phase.

The first phase is radical matching phase which is based on modified relaxation method to match each radical with templates in the radical database. Only templates that have the same number of strokes and cross-points as the extracted radical are selected for matching. The second phase is the matching with the knowledge database. All possible combinations of the recognized radicals are matched with the knowledge database. The recognized results after this phase are classified into four categories: "S", "M", "R", and "P". The character with status "R" or "P" needs to be processed in the third matching phase once more. The third phase is the matching of the whole character which is important for the proposed matching scheme as it promotes the recognition rate greatly.

The matching scheme proposed here is analogous to the one proposed in Wang et al. [3], except the difference in the first phase. The first phase is radical matching in

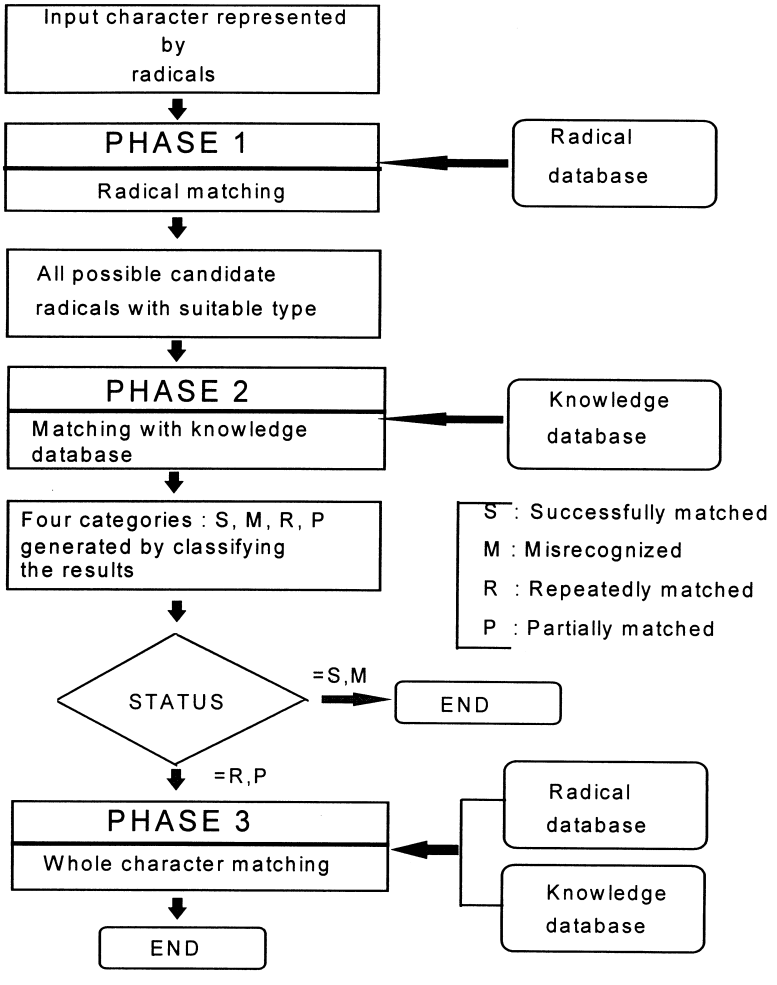

Fig. 17. The block diagram of proposed hierarchical radical matching scheme.

the proposed scheme, whereas it is partial matching in the one proposed previously.

5.1. *Phase 1*: *radical matching*

In this subsection, we will illustrate how to perform radical matching with the templates in radical database. The matching procedure is carried out by utilizing the modified relaxation scheme proposed by Wang et al. [19].

Shown in Table 1 are 32 radicals with some suitable and possible appearing types. Each radical has been observed and analyzed carefully to see whether it appears in the Chinese characters in Chi Hai (辭海). If the radical indeed appears in Chinese characters, then the corresponding type where the radical appears, are marked '**x**' in Table 1. From Table 1, we can see that some radicals may appear only in a few types, not all 19 types. For example, the radical "扌" (numbered 27 in Table 1) only appears in types 3, 12, 14, and 15. Shown in Table 2 is the appearing frequency in each type of 32 radicals for 300 test samples.

The matching scheme will generate a candidate radical set. In order to reduce the number of radical templates for matching and avoid making mistake, two rules are devised and stated as follows:

Rule 1: An input radical only needs to match with the templates whose number of strokes and number of cross-points are both the same as those of the input radical.

Table 1
Thirty-two radicals with order number and possible appearing types (marked by **x**)

Radical	No. of cross points	No. of strokes	Types appearing																		
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0 丁	0	2	x		x	x	x						x	x	x			x			
1 亻	0	2				x		x									x	x			
2 灬	0	3		x	x		x	x	x	x	x	x	x	x	x				x		
3 氵	0	3				x		x							x		x	x			
4 口	0	4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x		
5 火	0	4	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x		
6 阝	0	5				x	x								x		x	x	x		
7 月	0	5	x		x	x	x			x	x		x	x	x	x	x	x	x		
8 日	0	5	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
9 立	0	5	x	x	x	x	x	x	x		x	x	x	x	x				x		
10 方	0	5	x		x	x	x	x	x		x		x	x	x		x	x			
11 疒	0	5																			x
12 白	0	6	x	x	x	x	x	x	x	x	x	x	x	x	x		x		x		
13 辶	0	6																		x	
14 目	0	6	x		x	x	x	x	x	x	x		x	x	x		x		x		
15 貝	0	8	x	x	x	x	x			x	x	x	x	x	x		x		x		
16 十	1	2	x	x	x	x	x	x	x	x	x	x	x	x	x						
17 土	1	3	x	x	x	x	x	x	x	x	x	x	x	x	x		x				
18 寸	1	3	x		x	x	x				x		x						x		
19 千	1	3	x	x	x		x	x	x			x	x	x					x		
20 力	1	3	x		x	x	x	x	x	x	x		x	x	x		x	x	x		
21 木	1	4	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x		
22 王	1	4	x		x	x	x	x	x	x	x		x	x	x		x		x		
23 五	1	5	x	x			x		x			x	x								
24 耳	1	6	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x		
25 有	1	7	x		x	x	x		x				x	x	x		x	x	x		
26 金	1	8	x		x	x	x			x	x	x		x	x		x	x			
27 扌	2	3				x									x		x	x			
28 主	2	4		x				x	x			x									
29 甘	2	5	x	x			x					x						x	x		
30 由	2	6	x	x	x		x					x	x	x							
31 女	3	4	x	x	x	x	x	x		x	x	x	x	x	x		x	x			

Table 2
Appearing frequency in each type of 32 radicals for 300 test samples

Radical	total	Types frequency																		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0 丁	9					8											1			
1 亻	32				23		1									1	7			
2 灬	18						1	8	1	4	3		1							
3 氵	38				35											3				
4 口	128	6	1			7	1	16	6	40	4	10	9	25	3					
5 火	23		1	2	5	2	1	5	1	5			1							
6 阝	23				9	9	1										1	3		
7 月	32			1	4	8				9			1		2		3	4		
8 日	46		5	2	1	1	1	10	1	10	4	1	7	3						
9 立	32		1	1	1	4	1	19		1	2							2		
10 方	10					7	1	1												
11 疒	7																			7
12 白	19		2		1	7		7				1						1		
13 辶	5																		5	
14 目	10				4	2			1	1				1				1		
15 貝	7			1		2				4										
16 十	29			2		4	1	8	1	6	5		2							
17 土	58		2	2	6	4	5	19	2	11	1		3	1		2				
18 寸	13			1		4				3								5		
19 千	10					3	1	5			1									
20 力	19				1	6	1			2			1				7	1		
21 木	64	2	2	4	23	4	3	4	1	6	1	1	1			4	5	3		
22 王	25			1	6	4			1	7			2	1		3				
23 五	12					2		8			2									
24 耳	13				2	3	1		1	3	1						2			
25 有	7				1	4		1									1			
26 金	27				21	3										3				
27 扌	28				22									1		3	2			
28 主	8		1					6			1									
29 甘	11		1		1	5		3									1			
30 由	6					6														
31 女	17			2	9	2				2							2			

Rule 2: An input radical only needs to match with the templates which appear in the same location as the input radical.

Shown in Fig. 18 is an example illustrating this matching scheme. Fig. 18(a) is an input character "括". Fig. 18(b) is the extracted radicals which are "扌" with type 3, "千" with type 6, and "口" with type 8. Fig. 18(c) is the templates selected from the radical database for matching with the extracted radical. The radical "千" with type 6 only needs to match with templates "土", "千", and "力" but doesn't need to match with template "寸", since the template "寸" doesn't appear in type 6 location (see Table 1). Finally, we will obtain a candidate set which can be represented as {(扌, 3), (千, 6), (口, 8), …} as shown in Fig. 18(d).

5.2. *Phase 2*: *matching with knowledge database*

For each character to be recognized, the radicals and types it contains have to be determined by human first. For example, the character "抽" contains the radical "扌" (number 27 and type 3) and the radical "由" (number 30 and type 4). And then, collect them to establish knowledge database. The database is implemented as a double link list as shown in Fig. 19.

In order to perform the matching efficiently, a two-dimensional lookup table is built with its row and

column representing the number of strokes and number of cross points, respectively. Element (i, j) in the lookup table points to a link list which contains characters with i strokes and j cross points. The format of the lookup table is shown in Table 3. For example, the list of cell (9,4) contains two characters ("抽" and "挂").

Now, we can proceed the template matching task as described below. For an unknown input character, we have had a candidate radical set obtained from the previous radical matching process. The candidate radical set will be compared with those characters having the same number of strokes and number of cross points. All possible combinations of the recognized radicals are matched with knowledge database. For example, the input character "抽" which has 9 strokes and 4 cross points, will be compared with two characters "抽" and "挂". After this matching process, the recognized results will be automatically classified into four kinds of status:

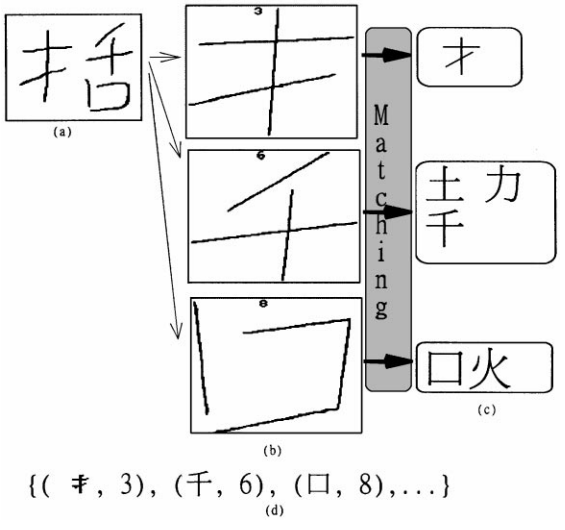

Fig. 18. (a) The input character "括". (b) The three extracted radicals after radical extraction. (c) The templates with which each extracted radical will match. (d) The candidate set after radical matching.

"S" The input character is successfully matched.
"M" The input character is misrecognized or not recognized.
"R" The input character is repeatedly recognized, that is, it is recognized as more than one candidate characters.
"P" The input character is partially matched, i.e., some radicals are not recognized.

An input character is decided as misrecognized if it is not recognized as any character or any radical in it. If the status of the recognized result is "R" or "P", then the input character needs to be processed once more in the next matching phase.

5.3. *Phase 3*: *whole character matching*

In the previous matching process, the input character with the status "R" or "P" is recognized as more than one candidate characters. To ensure which one is indeed the input character, the whole character matching process needs to be proceeded to resolve the ambiguity. We only establish all possible radical database and knowledge

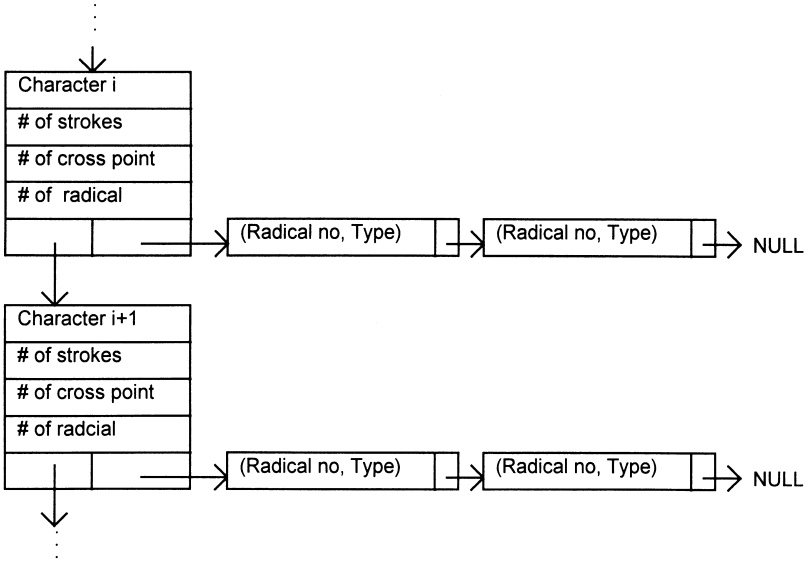

Fig. 19. The structure of knowledge database.

Table 3
The format of lookup table. The minimal and maximal number of stroke in our sample data are 4 and 17, respectively. The maximal number of cross-points is 6. The element (i, j) points to a link list which contains characters with i strokes and j cross-points

# of stroke \ # of cross point	0	1	2	3	4	5	6
4	(4,0)	(4,1)	(4,2)	(4,3)	(4.4)	(4,5)	(4,6)
5	(5,0)	(5,1)	(5,2)	(5,3)	(5.4)	(5,5)	(5,6)
6	(6,0)	(6,1)	(6,2)	(6,3)	(6.4)	(6,5)	(6,6)
7	(7,0)	(7,1)	(7,2)	(7,3)	(7.4)	(7,5)	(7,6)
8	(8,0)	(8,1)	(8,2)	(8,3)	(8.4)	(8,5)	(8,6)
9	(9,0)	(9,1)	(9,2)	(9,3)	**(9.4)**	(9,5)	(9,6)
• • •							
15	(15,0)	(15,1)	(15,2)	(15,3)	(15.4)	(15,5)	(15,6)
16	(16,0)	(16,1)	(16,2)	(16,3)	(16.4)	(16,5)	(16,6)
17	(17,0)	(17,1)	(17,2)	(17,3)	(17.4)	(17,5)	(17,6)

database for the character to be recognized instead of establishing the whole character database. To perform the whole character matching, the feature vectors of a whole candidate character must be constructed firstly.

The whole character matching process can be described as follows. For each candidate character, find its constituting radicals and associating radical types from knowledge database. Then the feature vectors of each radical with the associating type are obtained from the radical database and via proper transformations as defined in Wang et al. [3]. The feature vectors of the constituting radicals are combined thereafter to form the feature vectors of the whole candidate character. Last, the whole input character is matched with all candidate characters by making use of the modified relaxation method. The candidate character with the minimum distance is the recognized result. The whole character matching process is important for the hierarchical matching scheme as it will promote the recognition rate greatly.

An example shown in Fig. 20 is used to illustrate how to match a whole input character "括" with the whole candidate character "拐". Fig. 20(a) is candidate character "拐" with its knowledge came from knowledge database. Fig. 20(b) is the three constituting radicals obtained from radical database, Fig. 20(c) is the character which is constructed by combining the three radicals in Fig. 20(b) after suitable transformation, and Fig. 20(d) is the input character "括" which will match with the generated candidate character "拐" in Fig. 20(c).

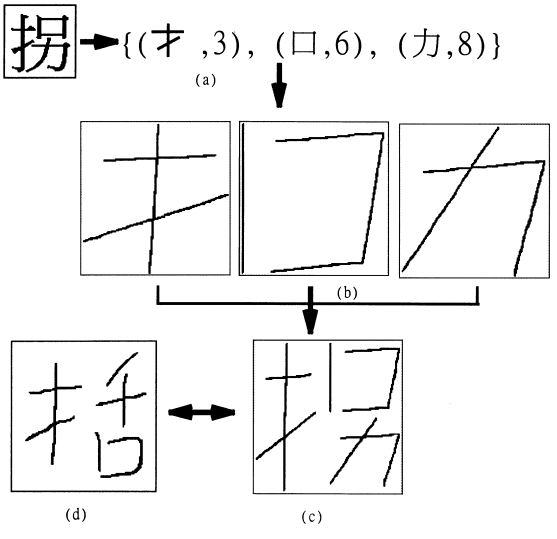

Fig. 20. Illustration of the whole character matching "括" with "拐".

6. Experimental results

The experiments were conducted on IBM PC-486 (DX-33) computer using C language. Two sample sets are collected for testing. The first sample set is shown in Fig. 21 which contains 300 characters with each character consisting of two to five radicals (as shown in Fig. 22) and had been written (under constraint) by six writers to form totally 1800 test samples. The second sample set as shown in Fig. 23 contains 156 characters.

Experiments for radical extraction were conducted on 1856 characters (the first sample and second sample sets). The successful rate of radical extraction is 92.5% as tabulated in Table 4. The average time for radical extraction is 0.65 s per character.

Experiments for matching process were conducted on 1800 characters (only the first sample set). The reason why we only use the first sample set for matching is that

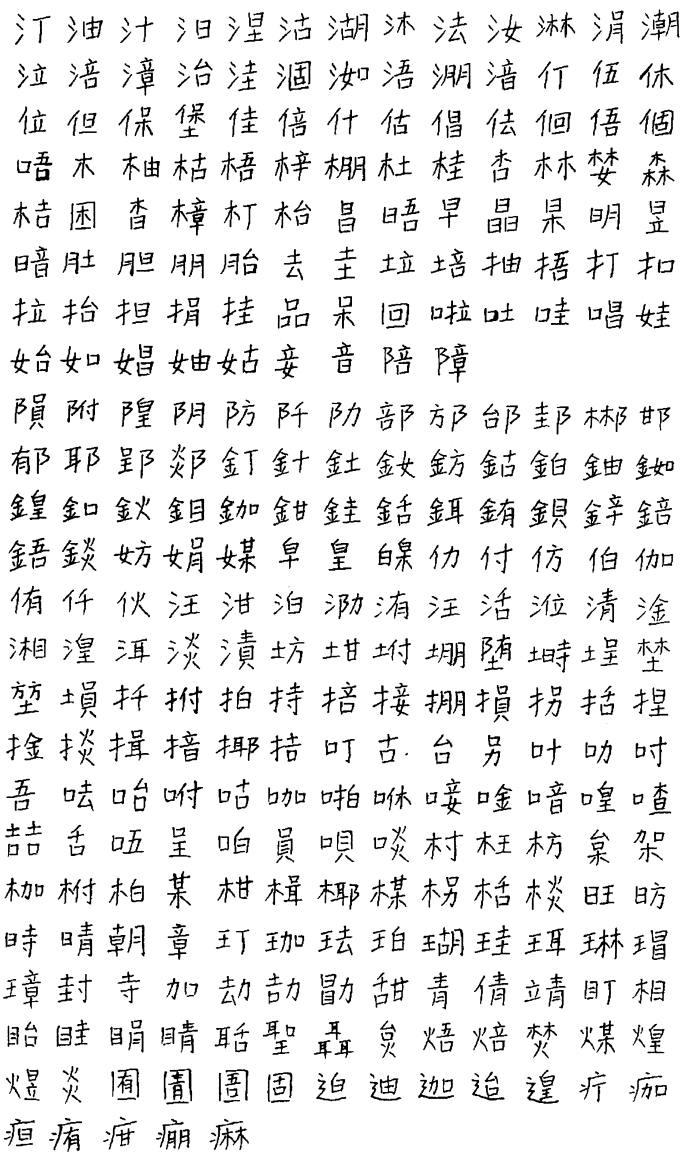

Fig. 21. The first sample set contains 300 characters which were written (under constraint) by six writers to form 1800 samples.

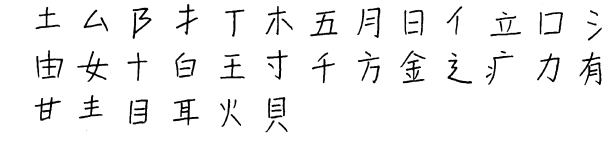

Fig. 22. The 32 radicals used in creating radical database.

the characters in the first sample set are composed of two to five radicals in 32 radical database. We treat the samples of the first three writers as the training set and the rest three samples as the testing set. The program and parameters are tuned to fit the training set. The overall recognition rate in our experiments is 98.2 and 80.9% as tabulated in Tables 5 and 6 (for training set and testing set, respectively). After the phase 3 (whole character) matching process, the recognition rate promote 11.5 and 27.2% as shown in Tables 5 and 6, respectively. This reveal that the whole character matching phase is important for our matching scheme.

The average recognition time of our hierarchical radical matching scheme is 0.274 second as tabulated in Table 7. There are totally 4716 radicals in 1800 characters. In average, one character consists of 2.62 radicals. Each character will match 7.28 radical templates in average. Thus, each radical will match 2.77 radical temples.

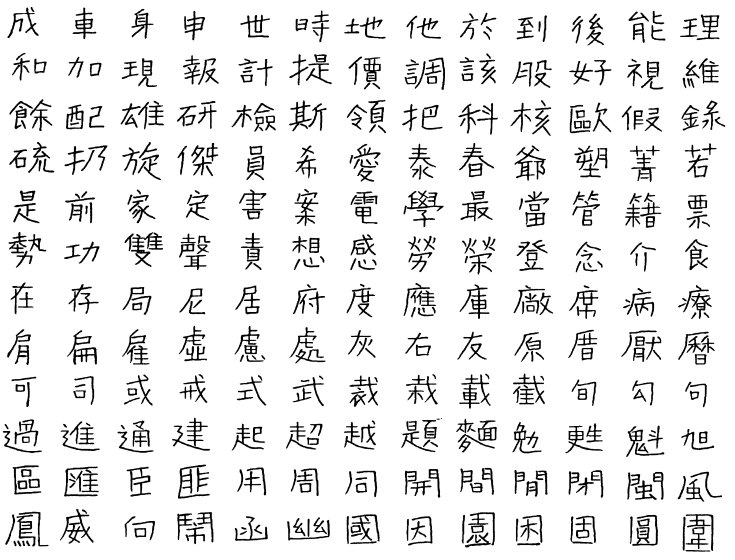

Fig. 23. The second sample set contains 156 characters.

Table 4
The tabulated results of radical extraction

Sample	1	2	Total
No. of characters	1800	156	1956
No. of success	1663	146	1809
Successful rate	92.4%	93.6%	92.5%

Table 5
The tabulated results of hierarchical radical matching scheme (for training set)

	Phase 1 + 2 matching		Phase 3 matching	
	Total	Ratio	Total	Ratio
Success	780	86.7%	884	98.2%
Mis-recognized	6	0.7%	16	1.8%
Repeatedly matched	20	2.2%	0	0.0%
Partially matched	94	10.4%	0	0.0%
Total	900	100%	900	100%

Table 6
The tabulated results of hierarchical radical matching scheme (for testing set)

	Phase 1 + 2 matching		Phase 3 matching	
	Total	Ratio	Total	Ratio
Success	483	53.7%	728	80.9%
Mis-recognized	71	7.9%	172	19.1%
Repeatedly matched	12	1.3%	0	0.0%
Partially matched	334	37.1%	0	0.0%
Total	900	100%	900	100%

Table 7
The recognition speed of hierarchical radical matching scheme

	Times needed on PC 486 DX-33 (Sec./Per Char.)
Phase 1	0.222
Phase 2	0.000
Phase 3	0.052
Total	0.274

There are two advantages of our proposed radical-based OCR system for handwritten Chinese character recognition:

1. After the radicals embedded in the characters with pattern, 4–10 have been extracted, they are identified immediately and need not to be recognized by radical matching.
2. Although some of the radicals embedded in a character may fail to be identified, the character can still be correctly recognized by our hierarchical radical matching scheme.

7. Conclusion

In this paper, a radical-based OCR system for handwritten Chinese character recognition is proposed. Since recognizing radicals is much easier than recognizing the whole character, the complexity of recognition will be greatly simplified if Chinese characters are decomposed into radicals before recognition. Hence a recursive hierarchical scheme is first proposed in module 1 to perform radical extraction. The proposed scheme, which combines structural method and statistical method, includes three layers. They are character pattern detection layer, straight cut-line detection layer, and stroke clustering layer. Next, character features and radical features are extracted in module 2. Last, a hierarchical radical matching scheme, which contains three phases, is proposed in module 3 to identify radicals embedded in an input Chinese character and then recognize the input character accordingly.

Experiments for radical extraction are conducted on 1856 characters. The successful rate of radical extraction is 92.5%. The average time for radical extraction is 0.65 s per character.

Experiments for matching process are conducted on two sets: training set and testing set, each set includes 900 characters. The overall recognition rate in our experiments is 98.2 and 80.9% (for training set and testing set, respectively). The average recognition time of our hierarchical radical matching scheme is 0.274 s. There are totally 4716 radicals in 1800 characters. In average, one character consists of 2.62 radicals. Each character will match 7.28 radical templates in average. Thus, each radical will match 2.77 radical temples. The experimental results reveal that our proposed method is feasible, flexible, and effective.

References

[1] R. Casey, G. Nagy, Recognition of printed Chinese characters, IEEE Trans. Electron. Comput. EC-15 (1966) 91–101.

[2] H. Ogawa, K. Taniguchi, Extraction of partial pattern in hand-printed Chinese character by relaxation, Trans. IECE Japan TGPRL80-40, 1980.

[3] A.B. Wang, J.S. Huang, K.C. Fan, Optical recognition of handwritten Chinese characters by hierarchical matching, J. Comput. Process. Chinese Oriental Languages 8 (2) (1994) 193–210.

[4] N. Babaguchi, T. Aibara, H. Sanada, K.Tezuka, Identification and extraction of radical from handprinted KANJI characters by segment correspondence method. Trans. IECE Japan PRL83-59, 1983.

[5] N. Tanaka, H. Aota, H. Sanada, Y. Tezuka, M. Shiono, A method of sub pattern extraction from handprinted KANJI characters, Trans. IECE Japan PRL83-26, 1983.

[6] B.S. Jeng, C.H. Lin, G.W. Yang, C.K. Lin, Automatic radical extraction system for machine-printed Chinese characters, Proceedings of NCS 1985, pp. 1216–1224.

[7] F.H. Cheng, W.H. Hsu, Partial pattern extraction and matching algorithm for Chinese characters. Proceedings of the International Conference on IASTED Robotics and Automation '85, 1985, pp. 18–22.

[8] F.H. Cheng, W.H. Hsu, Radical extraction from handwritten Chinese characters by background thinning method, Trans. IEICE, E 71(1) (1988).

[9] J.W. Chang, H.C. Liu, Radical extraction method for optical Chinese character recognition, Master Thesis, Institute of Computer Engineering, TamKang University, 1989.

[10] J.S. Lee, On-line Chinese character recognition system, Master Thesis, Institute of Electrical Engineering, Tsing-Hua University, 1987.

[11] C.W. Liao, J.S. Huang, A transformation invariant matching algorithm for handwritten Chinese character recognition, Pattern Recognition 23 (11) (1990) 1167–1188.

[12] M. Zhao, Two-dimensional extended attribute grammar method for the recognition of hand-printed Chinese characters, Pattern recognition 23 (7) (1990) 685–695.

[13] S.W. Lu, Y. Ren, C.Y. Suen, Hierarchical attributed graph representation and recognition of handwritten Chinese characters, Pattern Recognition 24 (7) (1992) 617–632.

[14] J.B. Kuo, M.W. Mao, A radical-partitioned coded block adaptive neural network structure for large-volume Chinese characters recognition, Proceedings of IJCNN, 1992, pp. 7–11.

[15] S.C. Lee, Z. Chen, Automatic Chinese character stroke ordering and its use in sub character decomposition, Master Thesis, Institute of Computer Science and Information Engineering, National Chiao Tung University, 1993.

[16] R.H. Cheng, Z. Chen, A knowledge based radical recognition method for handprinted Chinese character, Proceedings of International Conference on CVGIP, Taiwan, 1993.

[17] Y.H. Tseng, H.J. Lee, Knowldege-based radical extraction for handwritten Chinese characters, Master Thesis, Institute of Computer Science and Information Engineering, National Chiao Tung University, 1994.

[18] C.C. Hsieh, Model-guided recognition of handwritten Chinese characters, Ph. D. Thesis, National Chiao Tung University, Taiwan, ROC, 1992.

[19] A.B. Wang, J.S. Huang, K.C. Fan, Optical recognition of handwritten Chinese characters by the Modified relaxation, Image Vision Computing 12 (8) (1994) 509–522.

[20] C.C. Han, K.C. Fan, Y.L. Tseng, A.B. Wang, Coarse classification of Chinese characters via stroke clustering method, Pattern Recognition Lett. 16 (10) (1995) 1079–1089.

[21] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading MA, 1974.

About the Author—AN-BANG WANG was born in Kaoshiung, Taiwan, on 15 April 1955. He received his B.S. degree in Coorperative Economics from TamKang University, Taiwan, in 1977, the M.S. degree in Mathematics from Tamkang University, Taiwan, in 1980, and the Ph.D. degree in Computer science and Information Engineering from National Central University in 1996. At the same time, he joined the department of Information Management at Private Takming Business Junior College where he became vice-professor. In 1998, he joined the department of Banking and Insurance at Private Takming Business Junior College. Currently, he become the Dean of evening division at Private Takming Business Junior College. His current research interests include image analysis and optical character recognition.

About the Author—KUO-CHIN FAN was born in Hsinchu, Taiwan, on 21 June 1959. He received his B.S. degree in Electrical Engineering from National Tsing-Hua University, Taiwan, in 1981. In 1983 he worked for the Electronic Research and Service Organization (ERSO), Taiwan, as a Computer Engineer. He started his graduate studies in Electrical Engineering at the University of Florida in 1984 and received the M.S. and Ph.D. degrees in 1985 and 1989, respectively. From 1984 to 1989 he was a Research Assistant in the Center for Information Research at University of Florida. In 1989, he joined the Institute of Computer Science and Information Engineering at National Central University where he became professor in 1994. He was the chairman of the department during 1994–1997. Currently, he is the director of Software Research Center at National Central University. Professor Fan is a member of IEEE, and a member of SPIE. His current research interests include image analysis, pattern recognition, and computer vision.