Structural Boundary Feature Extraction for Printed Character Recognition

Jianming Hu, Donggang Yu, and Hong Yan

Department of Electrical Engineering
The University of Sydney, NSW 2006, Australia
{jianming, yu, yan}@ee.usyd.edu.au

Abstract. In this paper, an algorithm is developed for printed character recognition based on boundary analysis. New boundary smoothing schemes have been proposed, which can reduce noise significantly. The extracted boundary features are invariant to character size and are convenient for recognition. The whole algorithm is based on the chain code of each boundary and no floating operation is involved. The algorithm is tested with sample pages of a phone directory. Experimental results show that this algorithm is highly reliable and very fast.

1 Introduction

Character recognition has been studied extensively for a long time [1]-[2]. The research topics include both printed and handwritten character recognition [3]-[7]. Since printed characters have much less variances than handwritten characters, it is easier to design a recognition algorithm with very high recognition and reliability rates at low cost.

The aim of this paper is to design a recognition algorithm to read a phone directory. There are 68 symbols with various sizes and fonts to be recognized: upper and lower case of 26 English characters, 10 digits, and the symbols "&", "/", "(", ")", "+", "-". In the previous approach [7], we developed a recognition algorithm based on the decomposition and description of the skeleton of a character. Since the small bulbs of characters "r", "t", "f" are often removed by a thinning algorithm, and since the skeletons of "s" and "5", "g" and "9", "O (0)" and "D", bold "V" and "Y" are easily confused, these characters have to be verified further by analyzing their boundaries.

In this study we present a new structural boundary analysis method to recognize the characters with different sizes and fonts. Since we use a new efficient boundary smoothing algorithm to smooth the boundaries of each character, the smoothed boundaries reduce noise significantly and facilitate feature extraction.

2 Boundary Smoothing

In [8] a boundary smoothing algorithm for binary images was proposed, where the stroke width of a character is required to be at least 4 pixels. Since the stroke

width of a character in this application may be less than three pixels, we use some of the smoothing rules to avoid deleting small strokes.

Suppose a boundary P consists of N points $P = p_0, p_1, ..., p_i, ..., p_{N-1}$, where p_{i+1} is a neighbor of p_i $(i = 0, 1, ..., N-1)$ and the index i is calculated modulo N. Its chain code is given by $\overrightarrow{c_i} = \overrightarrow{p_i p_{i+1}}$. The following three rules are then used to smooth one boundary point each time:

Rule A: For three consecutive points p_{i-1}, p_i, and p_{i+1}, if p_i is a spurious point, ie., if the angle between segment $p_i p_{i+1}$ and $p_{i-1} p_i$ is 45^o (this can be determined from the chain codes c_{i-1} and c_i), then p_i is deleted.

Rule B: For three consecutive points p_{i-1}, p_i, and p_{i+1}, if $|c_i - c_{i-1}| = 4$, ie., if p_{i-1} and p_{i+1} correspond to the same point in the image, then p_i is deleted.

Rule C: For three consecutive points p_{i-1}, p_i and p_{i+1}, if c_i=1 (mod 2) and the angle between segments $p_i p_{i+1}$ and $p_{i-1} p_i$ is 90^o (this can also be determined from the chain codes c_{i-1} and c_i), then p_i is replaced by the neighboring point which lies in the line $p_{i-1} p_{i+1}$.

An example is shown in Fig. 1. To smooth the boundary further, we need to smooth two or more points at the same time. We first define a boundary primitive as a set of consecutive points which lie in a line. In a primitive all the points except the last one have the same chain code, which is defined as the direction of the primitive.

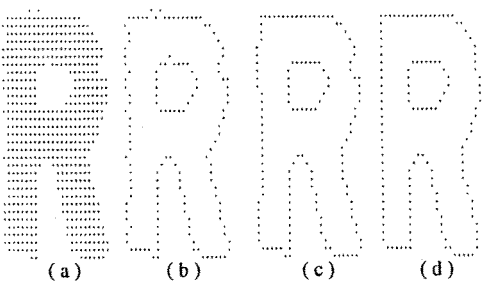

Fig. 1. Smoothing boundaries: (a) original image; (b) the boundaries before smoothing; (c) the boundaries after Rules A – C; (d) the boundaries after Rules D – F.

Suppose that there are M primitives in a boundary, then the boundary code $bc[i]$ corresponding to primitive i is defined as follows:

$$bc[i] = (s_i, t_i, l_i, dir_i) \quad (i = 0, 1, 2, ..., M) \tag{1}$$

where s_i and t_i are the positions of the starting and ending points of primitive i in the boundary chain, respectively; l_i is the length of the primitive; and dir_i is the direction of the primitive.

To determine which boundary primitive can be smoothed, we compare each primitive with its two neighboring primitives. The following three rules are used for smoothing multiple points.

Rule D: For primitive i, if the direction of primitives $i-1$, i, $i+1$ is one of the patterns shown in Fig. 2(a), and if one of the following conditions is true, then it can be smoothed.

(a) $l_{i-1}=1$, $l_{i+1}=1$, $dir_{i-2}=dir_i$, $dir_{i+2}=dir_i$, $l_i \leq l_{i-2}+l_{i+2}$;

(b) $l_{i-1}=1$, $l_{i+1}>1$, $l_i<l_{i-2}$, $dir_{i-2}=dir_i$;

(c) $l_{i+1}=1$, $l_{i-1}>1$, $l_i<l_{i+2}$, $dir_{i+2}=dir_i$.

Rule E: For primitive i, if the direction of primitives $i-2$, $i-1$, i, $i+1$, and $i+2$ match one of the patterns shown in Fig. 2(b), and if one of the following conditions is true, then it can be smoothed.

(a) $l_{i-1}=1$, $l_{i+1}=1$, $dir_{i-2}=dir_i$, $dir_{i-1}=1 \pmod 2$, $l_i \leq l_{i-2}$;

(b) $l_{i-1}=1$, $l_{i+1}=1$, $dir_{i+2}=dir_i$, $dir_{i-1}=1 \pmod 2$, $l_i \leq l_{i+2}$.

Rule F: For primitive i, if the direction of primitives $i-2$, $i-1$, i, $i+1$, and $i+2$ match one of the 16 patterns shown in Fig. 2(c), and if one of the following conditions is true, then it can be smoothed.

(a) $l_{i-1}=1$, $dir_{i-2}=dir_i$, $dir_{i-1}=1 \pmod 2$, $l_i \leq l_{i-2}$;

(b) $l_{i+1}=1$, $dir_{i+2}=dir_i$, $dir_{i-1}=0 \pmod 2$, $l_i \leq l_{i+2}$.

The point which has a small circle on it is deleted, and the other points are adjusted so that the smoothed boundary segment is the dotted lines. In the smoothing process we search and smooth the shortest primitive which can be smoothed each time. An example is shown in Fig. 1(d), which is the result of Fig. 1(c) after applying the multiple point smoothing algorithm.

3 Feature Extraction

Two kinds of features are extracted: the "main feature vectors" and the "supplementary feature vectors". A "global code" is used to characterize the global information of each boundary.

If the directions of primitive $i-1$, i, and $i+1$ match one of the patterns in Fig. 3, then primitive i is a feature vector. If the boundary segment goes in the counterclockwise direction, then it is labeled as a "v" (convex) vector. Otherwise it is labeled as an "a" (concave) vector. In this figure, primitives $i-1$ and $i+1$ may be either of the two primitives denoted by the dashed lines. The corresponding main feature vector is defined as follows:

$$mv(j) = (label, s, t, dir, l) \tag{2}$$

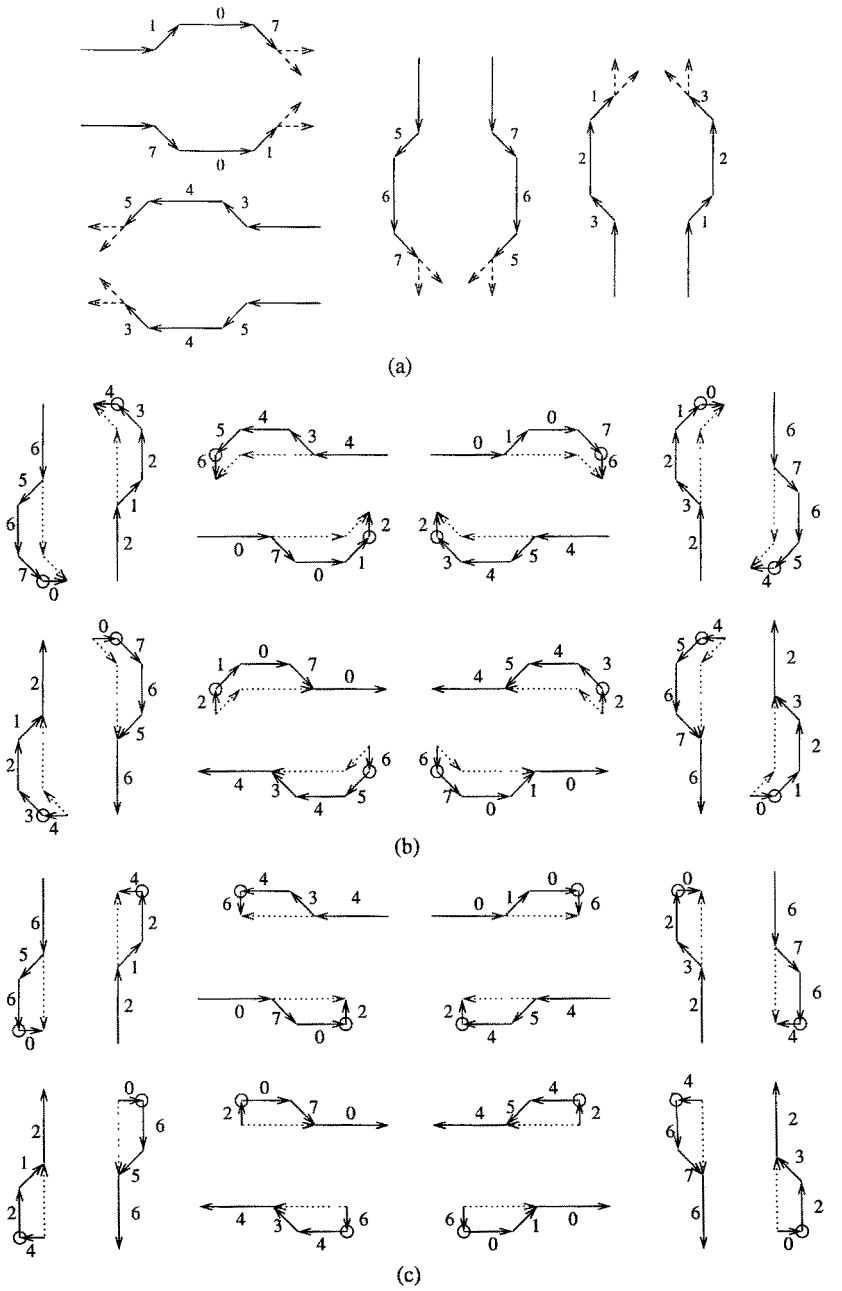

Fig. 2. The models used in the multiple point smoothing algorithm: (a) corresponding to Rule D; (b) corresponding to Rule E; (c) corresponding to Rule F.

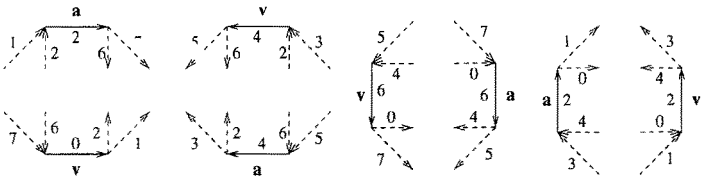

Fig. 3. Patterns for the extraction of main feature vectors.

where j is the order of the feature vectors for a boundary; "label" is "v" or "a"; "s" and "t" are the positions of the starting point and end point of the primitive, respectively; "dir" is the direction of the primitive, and "l" is the length of the primitive.

The global code for each boundary is defined as a 5-dimensional vector (g_1, g_2, g_3, g_4, g_5), where g_1 and g_2 are the numbers of "v" and "a" feature vectors, respectively, g_3 to g_5 are the numbers of feature vectors whose directions are chain codes 0, 2, 4, and 6, respectively. Figure 4 shows some examples of the extracted main features.

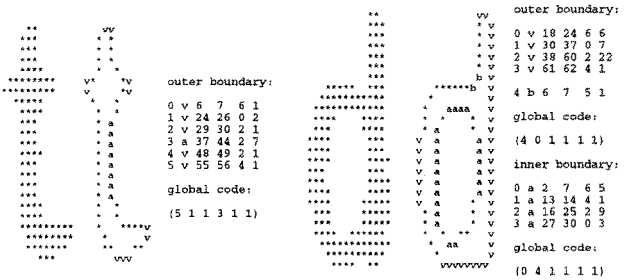

Fig. 4. Some sample characters, their smoothed boundaries and the extracted feature vectors.

The supplementary feature vectors are used to characterize a boundary segment where a bending happens. The "b" feature vector is used to detect a boundary primitive where the boundary changes its direction by 90^o, and the "c" feature point is used to detect a curve change by more than 90^o. Figure 5 shows the detection patterns, where the length of the "c" feature vector is set to 0.

The detection of "b" and "c" feature vectors is applied to the boundary segments between two main feature vectors only. Figure 4 has shown some examples, where the supplementary feature vectors are listed after the main feature vectors and which are not counted in the global codes.

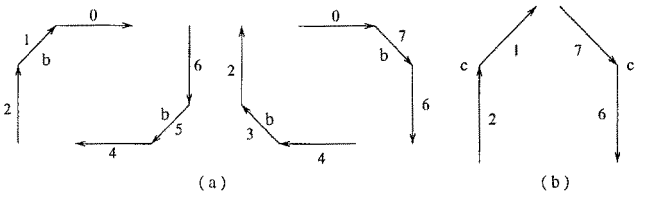

Fig. 5. Patterns for the extraction of supplementary feature vectors: (a) for "b" feature vectors; (b) for "c" feature points.

4 Recognition

4.1 Character Models

The models for each character are constructed according to its possible main feature vectors. Figure 6(a) is the feature model for character "M". Due to noise and the smoothing algorithms, some characters will have two or more models. For each of these characters the standard model and the possible deformation models are considered. Figure 6(b) shows an example of the standard model of character "d" and its three possible deformation models. Note that a deformation model is formed by reducing one or more pairs of neighboring "v" and "a" feature vectors.

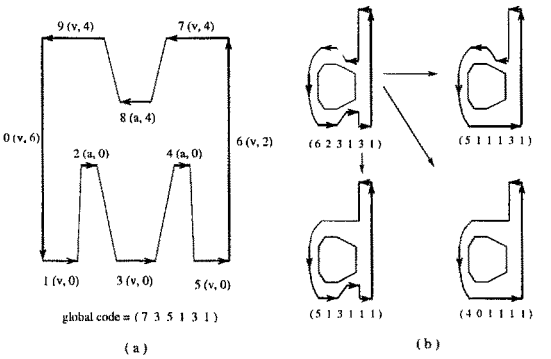

Fig. 6. Character models: (a) the model for character "M"; (b) the standard model and its possible deformation models of character "d".

4.2 Matching

An input character is first passed to the corresponding subclass according to its global code, and then compared with the models in that subclass. In the matching procedure, we first search for the starting main feature vector, which

is specified as the first "v" vector whose direction is chain code 6 in the leftmost part of the image. The extracted feature vectors of the input character are then compared with the descriptions of the feature vectors of the character models.

For most characters we need only to match the descriptions of the main feature vectors. For characters "F", "L", "P", "T", "b", "d", "h", "p", "q", "1", "4", and "+", we need further to verify if there is a "b" feature vector in the corresponding boundary segment. For the recognition of character "Y", the detection of "c" feature points is performed.

4.3 Repairing

When a small hole (bump) is more than one pixel in depth (height) and more than one pixel in width, it cannot be smoothed by the one point and multiple point smoothing algorithms. Figure 7 shows some examples. Therefore, we repair these boundaries using the extracted main feature vectors and the boundary codes.

Fig. 7. Examples of characters for repairing.

The repairing patterns in the horizontal direction are shown in Fig. 8, where the right pattern of each group is the repaired result of the left pattern. The repairing patterns in the vertical direction can be similarly described.

5 Experimental Results

The segmentation algorithm has been discussed in [7], where the images are scanned at 400 dpi in binary mode. Since the algorithm described in this paper is for recognition only, both the upper cases and lower cases of the characters "c", "o", "p", "s", "u", "v", "w", "x" and "z" are recognized as lower cases. Character "O(o)" and numeral "0" are all recognized as "o". These characters can be distinguished by the contextual information (layout of a group of characters, image size, etc. [7]). 10 pages selected from different parts of the phone directory were used for testing, each consisting of about 13,000 characters (not including "."). The results were 0.38% rejection rate and 99.62% recognition rate with

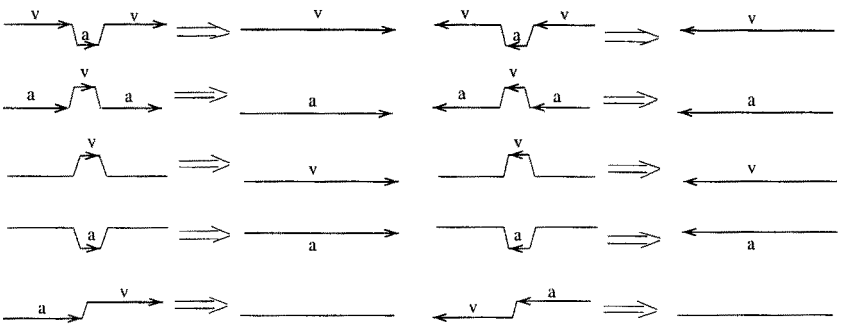

Fig. 8. Illustration of the repairing patterns in horizontal direction.

reliability over 99.99%, which improves the previous results based on skeleton analysis [7].

6 Conclusion

An algorithm is proposed for printed character recognition based on boundary analysis. The algorithm can recognize almost all the unbroken characters correctly and reject all the broken characters. In practical use a statistical classifier can be combined to recognize the broken characters in order to increase the overall recognition rate.

References

1. Govindan, V. K., Shivaprasad, A. P.: Character recognition – a review. Pattern Recognition **23** (1990) 671-683
2. Mori, S., Suen, C. Y., Yamamoto, K.: Historical review of OCR research and development. Proc. of the IEEE **80** (1992) 1029-1058
3. Suen, C. Y., Nadal, C., Legault, R., Mai, T. A., Lam, L.: Computer recognition of unconstrained handwritten numerals. Proc. of the IEEE **80** (1992) 1162-1180
4. Rocha J., Pavlidis, T.: A shape analysis model with applications to a character recognition system. IEEE Trans. Pattern Anal. Mach. Intell. **16** (1994) 393-404
5. Kahan, S., Pavlidis, T., Baird, H. S.: On the recognition of printed characters of any font and size. IEEE Trans. Pattern Anal. Mach. Intell. **9** (1987) 274-288
6. Pavlidis, T.: Algorithms for shape analysis of contours and waveforms. IEEE Trans. Pattern Anal. Mach. Intell. **2** (1980) 301-312
7. Hu, J., Yan, H.: Automatic reading of white pages in a telephone directory. Optical Engineering **35** (1996) 3150-3158
8. Yu, D., Yan, H.: An efficient algorithm for smoothing, linearization and detection of structural feature points of binary image contours. Pattern Recognition **30** (1997) 57-69