An Improved Approach to Generating Realistic Kanji Character Images from On-Line Characters and its Benefit to Off-line Recognition Performance

OndIej Velek [1], Cheng-Lin Liu [2], Stefan Jaeger [1], Masaki Nakagawa [1]
[1] *Tokyo University of Agri. & Tech. 2-24-16 Naka-cho, Koganei-shi, Tokyo 184-8588, Japan*
[2] *Central Research Laboratory, Hitachi Ltd., 1-280 Higashi-koigakubo, Kokubunji-shi, Tokyo, Japan*
E-mail:{velek,stefan}@hands.ei.tuat.ac.jp , liucl@crl.hitachi.co.jp, nakagawa@cc.tuat.ac.jp

Abstract

This paper proposes a method for generating realistic calligraphic Kanji character images from on-line data. The proposed method is an improvement of our former method presented in [1]. Our new method can cope also with connected on-line strokes, i.e., stroke number variations, which were not correctly painted in our previous method. The new method decomposes strokes into three different parts (end part, bend part, connecting part), and paints each part according to a prototypical shape assigned to it from a calligraphic stroke shape library.

Our generated calligraphic off-line images serve two purposes: First, they provide additional training samples for off-line recognition. Second, they allow application of off-line methods in on-line recognition. This paper also presents some experiments for these purposes.

1. Introduction

The availability of sufficient training patterns is an essential requirement for designing high-performance classifiers for character recognition. Collecting large quantities of genuine patterns is a time and money-consuming process (especially for large character sets of Oriental languages [2]), so that utilizing artificially generated training patterns represents an interesting, cheaper alternative.

A straightforward approach for generating artificial patterns is to apply geometrical transformations to genuine off-line images ([3],[4]). Another possibility is to decompose images into several logical components and to apply different transformations to each component based on a generative model. This generates new characters with different relational structures ([5]-[7]).

Our approach is to utilize existing on-line databases for off-line recognition. In our previous work [1], we introduced three different methods for generating bitmap character images from on-line data: constant line painting mode, proportional mode (stroke thickness depends on writing speed), and calligraphic mode. In calligraphic mode, we combine on-line data with a calligraphic stroke-shape library. This off-line library contains shapes of strokes written by writers using different writing tools. By combining stroke-patterns written by brush we can generate calligraphic characters for on-line data. Our previous calligraphic painting mode ([1]; 3.2) is based on primitive stroke identification (PSI). Each elementary stroke is classified and painted using the appropriate shape from the stroke-shape library. For strokes not assigned to any prototype in the library, a default stroke pattern is applied.

Section 2 of this paper introduces a new calligraphic mode. Section 3 demonstrates some experiments using patterns generated by our new method. Section 4 summarizes our results and discusses future work.

2. Calligraphic painting mode based on stroke components classification (SCC)

Our previous method PSI was based on recognition of stroke-types. A decomposition of off-line Kanji characters into strokes is natural since strokes are the basic components of Japanese and Chinese characters.

The discrete stroke type of a neatly written character can usually be successfully recognized, but when the character is written quickly or in some non-elementary calligraphic style, strokes are often connected and thus cannot be classified.

In our improved method, we solve this problem by dividing strokes into several parts. Each part is classified into one of three classes: “end parts”, “bend parts”, and “connecting parts” according to the following definitions:

(a) Each stroke has just two *end parts*: head and tail.
(b) The part with high curvature are called *bend parts*
(c) Bending and ending parts are connected to each other by so-called *connecting parts*

The stroke-shape defining parts are mainly end parts and bend parts because writing direction, pressure, or pen

posture often significantly changes in these parts. We detect end parts and bend parts of each stroke, classify them to appropriate subclasses, and paint them using a pattern from the stroke-shape library.

2.1. Bending points

2.1.1. Detection of bending points

To use local characteristics, such as the first and second derivation, is inappropriate, because strokes are not continuous functions and are often distorted by local noise. However, we can find bend parts of strokes in a way similar to detecting points with the maximum distance from a chord. This algorithm, which is described below, is robust against local noise and is used in several variations in many on-line recognition systems [8],[9].

(a) Initialize: A = head of stroke; B = tail of stroke

(b) Find Point C which has the maximum distance to the Chord AB

(c) If the distance d(C,AB) is higher than a given threshold, add Point C to the set of bending points and find the points with the maximum distance for Chord AC and Chord CB recursively {goto (b)}

Figure 1 shows several trajectories together with their recorded points and their bending points, which are boldly printed.

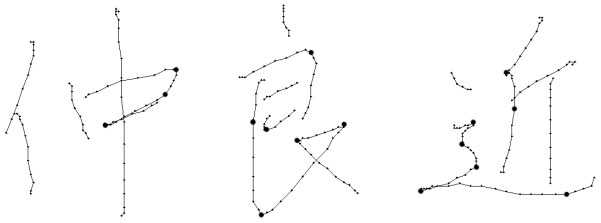

Figure 1. Detection of bending points.

2.1.2. Classification of bending points

We define four types of stroke bends as shown in Figure 2. Three corner bends S_a, S_b, S_c and one round bend R_{abc}. We don't need to define more than one type of round bends because stroke width is approximately the same for wide and narrow round bends when being written by brush.

These four types are sufficient to describe the most important differences of stroke width in bend parts.

Let each stroke be a sequence of $\underline{n}$ sampling points: $stroke=\{a_0; a_1;...; a_n\}$. Lets rename the indices of this sequence as follows: $stroke=\{A_0; a_{0,1};\ldots; a_{0,k0}; A_1; a_{1,1};\ldots; a_{1,k1}; \ldots ... A_{n-1},; a_{n-1,1};\ldots; a_{n-1,kn-1}; A_n\}$, where $A_x = a_{x,0}$; $x \in 0..n$; are recorded points detected as bending points or end points; $a_{x,y}$, $y \neq 0$, are all other sampling points. (Figure 3) Then each bending point is classified according to the following three features (one global, and two local features):

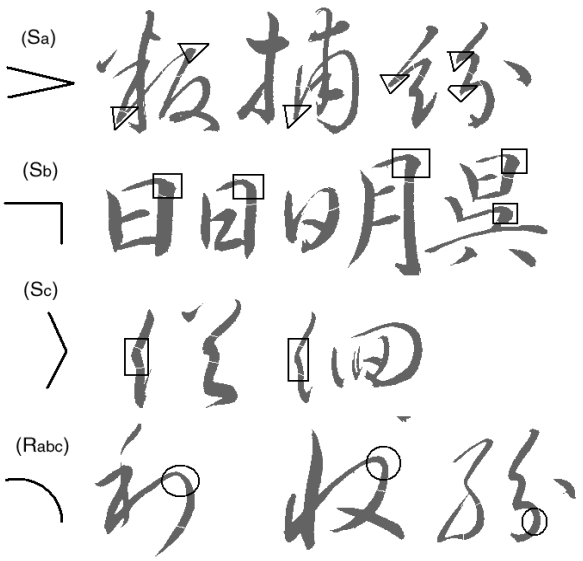

Figure 2. Examples of four bends. Corner bends (S_a) (S_b), (S_c) and round bend (R_{abc}).

(a) Global angle α: An angle subtended by neighboring bending points: $\alpha = (A_{i-1})(A_i)(A_{i+1})$, see (Figure 4.$\alpha$). This global feature distinguishes three types of corner bends (S_a) (S_b), (S_c).

(b) Local angle β: An angle subtended by the neighboring points $(a_{i-1,k})(A_i)(a_{i,1})$, as shown in Figure 4.$\beta$. This local feature distinguishes the corner bends (S_a) and (S_b) from a round bend (R_{abc}). However, it is highly sensitive to noise because the definition of angle β is based only on three sampling points. Moreover, this feature cannot distinguish a corner bend (S_c) from a round bend (R_{abc}) (An example follows in the next paragraph.)

(c) Local angles ω defined by the change of stroke direction in bending points, normalized to the change of direction in two neighboring bending points. $\omega = \frac{(a_{i-1,k_{i-1}-1})(a_{i-1,k_{i-1}})(A_i) + (A_i)(a_{i,1})(a_{i,2})}{2} - (a_{i-1,k_{i-1}})(A_i)(a_{i,1}) =$ $=(\beta_{i-1}/2+\beta_{i+1}/2)-\beta$; see Figure 4.$\omega$. If a bend is smooth (type (R_{abc}) from Figure 3) ω converge to zero, because for the round bend $\beta=\beta_{i-1}=\beta_{i+1} \Rightarrow \omega = 0°$. For a corner bend the value of an angle ω grows: $\omega \cong 180-\beta$. The difference between Feature (c) and Feature (b) is illustrated in Figure 4.ω. Feature (b) is the same for the smooth stroke sequence AXYZ and sharp sequence OXYZ ; $\beta=120°$; but Feature (c) differs: for OXYZ $\omega \cong 180-120=60°$, and for AXYZ $\omega \cong 0°$.

The recognition scheme using these three features for classifying into four classes is shown in Figure 6.

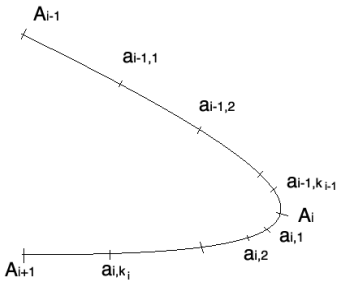

Figure 3. Stroke with three bending points: A_{i-1} ; A_i ; A_{i+1}.

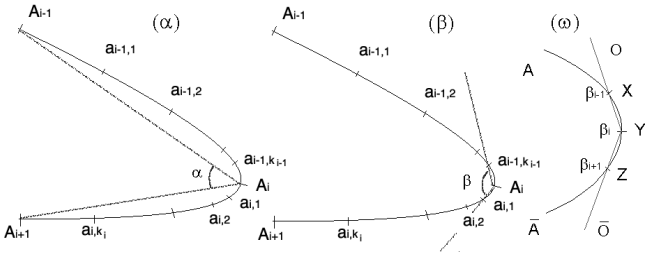

Figure 4. Graphical description of features (α), (β), (ω).

An example of classification based on feature vector v(α,β,ω) can be seen in Fig.5: All bending points of Fig.1 have been classified to the subclasses defined in Fig.2.

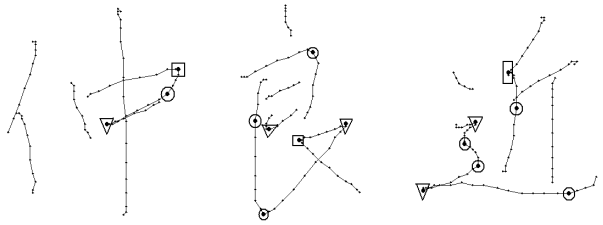

Figure 5. Classification of bending points to the subclasses defined in Figure 2. Symbols (square, circle, etc.) used for marking bending points are defined in Figure 2.

Patterns added to the stroke-shape library and used for bend parts can be seen in Figure 7 (original patterns) and Figure 8 (normalized patterns). More information on these patterns is given in Reference [1].

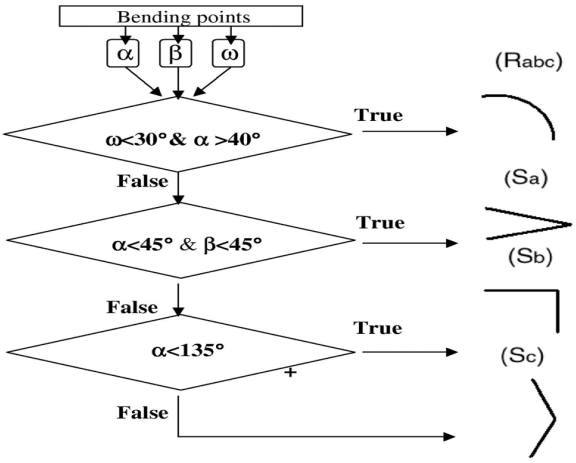

Figure 6. Recognition schemes for bending points.

Figure 7. Stroke image templates of bend parts.

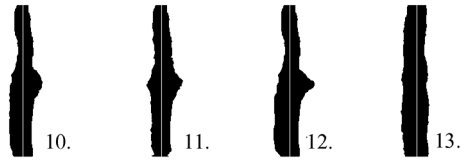

Figure 8. Normalized stroke images from Figure 7.

2.2. End points

Detection of end points is straightforward because they correspond to the heads and tails of strokes. We only need to recognize their types and to paint them using appropriate template-bitmaps from our stroke-shape library.

For classification of end points we use a slightly modified version of our algorithm for classifying entire strokes ([1], paragraph 3.2.3). The input to this algorithm, which is based on directional vectors, is the part from the ending point to the nearest bending point of a stroke. The recognized stroke type is always a straight type, because end points are directly connected with bending points. Depending on the end point (head or tail), we use the appropriate half of the library shape for painting.

Figure 9. Recognized end parts (boldly painted).

2.3. Comparison of SCC and PSI.

PSI should be preferred to SCC, because PSI is based on elementary strokes, which are natural components of Japanese and Chinese characters. If, however, strokes are connected and stroke recognition does not work, SCC should be used.

Figure 10. Strokes visualized by SCC.

3. Experiments.

Our generated patterns have two purposes: to provide more samples for off-line recognition, and to apply off-line methods to on-line recognition. The following section describes our experiments for underpinning the benefits of our methods regarding these purposes.

3.1. Generating samples for off-line recognition.

We generated four training sets from a subset of the Nakayosi on-line database [10], (using constant mode, proportional mode, PSI, and SCC). Each set contains 150 samples for 3036 categories respectively (2965 Kanji, 71 Hiragana). The size of the generated patterns is 96x96 pixels. The fifth training set contains real off-line patterns.

We trained an off-line recognizer with these five training sets. This recognizer represents each character as a 256-dimensional feature vector. Every input pattern is scaled to a 64x64 grid by nonlinear normalization and smoothed by a connectivity-preserving procedure. The normalized image is then decomposed into 4 contour sub-patterns, one for each direction. A 64-dimensional feature vector is extracted from each contour pattern as its convolution with a blurring mask (Gaussian filter).

A pre-classification step precedes the actual final recognition. Pre-classification selects 50 candidates with the shortest Euclidian distance between their mean vectors and the test pattern. The final classification uses a modified quadratic discriminant function (MQDF2), developed by Kimura [11] from traditional QDF.

We tested our recognizer trained with different data on two test sets containing real off-line patterns. Test1 is from ETL9b, Test2 from JEITA-HP database. Our achieved recognition rates are shown in Table 1.

Table 1. Recognition rates of real off-line data by training with generated data and genuine off-line data.

	Painting mode for on-line data				off-line
train	constant	proportion	PSI	SCC	---
test1	91.82	94.68	95.75	95.97	97.75
test2	88.78	92.63	93.61	93.89	95.53

We can see in this experiment that PSI and SCC outperform constant and proportional painting mode, because they generate samples more similar to real characters.

3.2. Off-line methods for on-line recognition.

In this experiment we divided each of four training sets (from experiment 3.1) into a training set (100 samples per category) and a test set (50 samples per category), so that training and test data were generated by the same method. In Table 2 we can see that the recognition rate is decreasing for sophisticated methods PSI, SCC, which generate more realistic patterns. One of the reasons for this is that extracting features from strokes of constant width is simpler than extracting information from more complicated shapes.

Table 2. Online recognition rates by off-line method with four painting modes

Mode	constant	proportional	PSI	SCC
Rec.rate	95.25	95.12	94.36	93.50

4. Conclusion

In this paper we introduced a new method for generating realistic character patterns. SCC can visualize also connected strokes, which is a problem for PSI. In an experimental section we proved that PSI & SCC generate more useful patterns for training off-line recognizers.

In our future work, we will combine on-line and off-line data in a training set and we will investigate how adding on-line data can improve recognition rate.

References

[1] O.Velek, C.-L. Liu, M.Nakagawa, Generating Realistic Kanji Character Images from On-line Patterns, *Proc. 6th ICDAR*, 2001, pp.556-560.

[2] T.H. Hilderbrand, W. Liu, Optical recognition of Chinese characters: advances since 1980, *Pattern Recognition*, 26(2), 1993, pp.205-225.

[3] T. Ha, H. Bunke, Off-line handwritten numeral recognition by perturbation method, *IEEE Trans. PAMI*, 19(5), 1997, pp.535-539.

[4] M. Mori, et al., Generating new samples from handwritten numerals based on point correspondence, *Proc. 7th IWFHR*, 2000, pp.281-290.

[5] H. Nagahashi, M. Nagatsuyama, A pattern description and generation method of structural characters, *IEEE Trans. PAMI*, 8(1), 1986, pp.112-117.

[6] C.-H. Tung, Y.-J. Chen, H.-J. Lee, Performance analysis of an OCR system via artificial handwritten Chinese character generation, *Pattern Recognition*, 27(2), 1994, pp.221-232.

[7] G. Ghosh, A.P. Shivaprasad, An analytic approach for generation of artificial hand-printed character database from given generative models, *Pattern Recognition*, 32(6), 1999, pp.907-920.

[8] U. Ramer, An interactive procedure for polygonal approximation of plane closed curves, *Computer Graphics and Image Processing 1*, 1972, 244-256.

[9] K. Ishigaki, T.Morishita, A top-down online handwritten character recognition method via the denotation of variation, *Proc .ICCPCOL.* 1998, pp.141-145

[10] M. Nakagawa, et al., On-line character pattern database sampled in a sequence of sentences without any writing instructions, *Proc. 4th ICDAR*, 1997, pp.376-380.

[11] F. Kimura, et al, Modified quadratic discriminant function and the application to Chinese characters, *IEEE Trans. PAMI 9(1)*, 1987, 149-153.