Vol. 44 No. 1 IPSJ Journal Jan. 2003

Regular Paper

Generation of Japanese Cursive Sentences Using Optimal Smoothing Splines

Hiroaki Nakata[†] and Hiroyuki Kano[†]

A problem of designing and generating cursive characters and sentences is considered. Based on the concept of "dynamic font", we developed an interactive font design system with various functions for designing and manipulating characters. Such a system provides us with an efficient development environment, and is used to produce Japanese cursive Kana characters. We then developed a smoothing operation on characters by employing the theory of optimal smoothing splines, which can be used for transforming their typefaces to the omitted running style. Some examples show that such an operation eventually generates naturally-smoothed cursive sentences.

1. Introduction

With a rising interest in desktop publishing (DTP), symbolized by the widespread use of *Nengajo* (New Year's card) printing softwares in recent years, a variety of characteristic Japanese brush-written fonts have been developed together with the designing tools. In general, the methods for generating fonts have been classified into three categories [1] as: i) dot matrix, ii) outline vector, and iii) skeleton vector font schemes. In DTP, dot matrix fonts (e.g., bitmap font) and outline vector fonts (e.g., TrueType) have been used for designing characters. The skeleton vector font scheme is believed to be more suited for designing brush-written characters, and some generation methods based on this scheme have been reported [2)~4)].

However, in these schemes, characters are designed essentially as some static planar patterns, and thus they are not very suited to further treatments such as concatenating characters or changing the typefaces systematically. In addition, if we consider the case of hand written characters, the characters are obtained obviously by moving a pen or some other writing device on the paper continuously in both time and space. In other words, they are obtained as the result of hand motions. It is thus significant to develop a scheme for generating and designing hand-written characters, Japanese cursive characters in particular, in a dynamical as well as systematic fashion.

Inspired by the concept of "unit motions" for synthesizing free motions of robotic arm [5)], the concept of "dynamic font" for designing fonts has been introduced [6)]. Here, the idea is that characters are generated by intersecting a virtual writing device with a virtual writing plane and moving the device continuously in both space and time according to the designed writing motion. Such a motion is synthesized by a time-shifted and weighted sum of normalized uniform B-spline functions [7)].

The purpose of this paper is

(I) to provide an efficient designing environment for dynamic fonts, and

(II) to generate brush-written Japanese cursive sentences in some systematic way.

In particular, we develop a font designing system, called font editor, together with a descriptive language for font data. Using the editor, we design cursive Kana characters and generate successive scripts. We also develop a novel re-designing scheme of characters by introducing smoothing operation on fonts based on the theory of optimal smoothing splines [8),9)]. It is shown that such a scheme transforms the typefaces of characters to the omitted running style, and eventually generates naturally-smoothed cursive sentences [10)~13)].

This paper is organized as follows. The model for generating characters is described in Section 2. Then, Section 3 presents the font design system, Section 4 shows a method for generating words or sentences, Section 5 introduces a smoothing operation, and finally, concluding remarks are given in Section 6.

2. Generation of Characters

2.1 Writing Motion and Characters

Figure 1 illustrates a model for generating

† Department of Information Sciences, Faculty of Science and Engineering, Tokyo Denki University

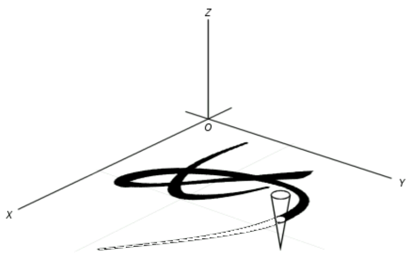


Fig. 1 Process of generating characters.

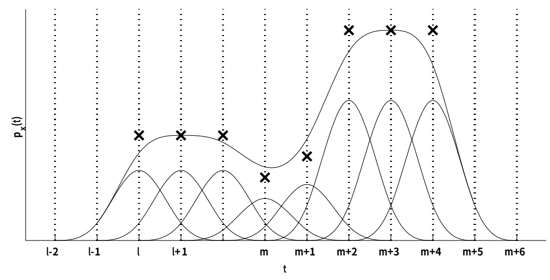


Fig. 2 Motion composed of unit motions.

characters[6]. A virtual writing plane is defined as 2-dimensional $O-XY$ plane, and a virtual writing device is defined as a circular cone with the axis parallel to Z-axis. The character is obtained by moving the device in 3-dimensional space $O-XYZ$ according to the designed writing motion and then by deriving the cross-sectional area on the plane. Hence, as the device moves downwards, the resulting stroke becomes thicker, and while it is not intersecting with the writing plane, nothing is printed.

The motion of the writing device tip, $p(t) \in \mathbf{R}^3$, is assumed to be given by time-shifted and weighted sum of B-splines[5] as,

$$p(t) = \sum_{i=l-k+1}^{m+k-1} p_i B_k(\alpha(t-t_i)). \qquad (1)$$

Here, l and $m\,(\geq l)$ are integers that determine the time interval of motion, p_i's are 3-dimensional weighting vectors called "control points", $B_k(\cdot)$ is a normalized uniform B-spline function of degree k with integer knot points, and α is a positive constant scalar employed for scaling the interval between equally-spaced knot points t_i with $t_{i+1}-t_i = 1/\alpha$. It is noted that the $B_k(\cdot)$ can be generated recursively by the following algorithm[7]:

$$B_k(t) = \begin{cases} N_{k-j,k}(t-j) & j \leq t < j+1 \\ & (j=0,\cdots,k) \\ 0 & t<0, k+1 \leq t, \end{cases} \qquad (2)$$

where $N_{j,k}(t)$ are the base functions derived by

$$\begin{aligned} N_{0,i}(t) &= \frac{1-t}{i} N_{0,i-1}(t) \\ N_{i,i}(t) &= \frac{t}{i} N_{i-1,i-1}(t) \\ N_{j,i}(t) &= \frac{i-j+t}{i} N_{j-1,i-1}(t) \\ &\quad + \frac{1+j-t}{i} N_{j,i-1}(t), \quad j=1,\cdots,i-1, \end{aligned}$$

for $i = 1,2,\cdots,k$ with the initial function $N_{0,0}(t)=1$. Cubic splines are most frequently used in practice, and $N_{j,3}(t)$ defining $B_3(t)$ is computed as

$$\begin{aligned} N_{3,3}(t) &= t^3/6 \\ N_{2,3}(t) &= (1+3t+3t^2-3t^3)/6 \\ N_{1,3}(t) &= (4-6t^2+3t^3)/6 \\ N_{0,3}(t) &= (1-t)^3/6. \end{aligned} \qquad (3)$$

Note that the three dimensional motion $\mathbf{1}B_k(\cdot)$, where $\mathbf{1} = (1\ 1\ 1)^T$, is called "unit motion", and the series of unit motions: $\mathbf{1}B_k(\alpha(t-t_{l-k+1})), \cdots, \mathbf{1}B_k(\alpha(t-t_{m+k-1}))$ form a basis having the local and minimal support for the space of spline functions of degree k. The motion $p(t)$ in (1), synthesized by taking a superposition of weighted unit motions, thus has the smoothness property of spline functions of degree k. **Figure 2** shows an example of motion composed of unit motions, where x-component of $p(t)$ is shown together with the control points ('cross' mark) for the case $k=3$ and $\alpha=1$.

The motion can be an arbitrary spline of degree k only in the interval $[t_{l+1},\ t_{m+k}]$, since here we have overlaps of $k+1$ B-splines. In this sense, we consider $[t_{l+1},\ t_{m+k}]$ as the time interval of specific interest of the motion, and $p(t)$ in such an interval is computed easily as

$$p(t) = \underline{p}\ A_k\ \underline{t}\ ,$$

where

$$\underline{p} = \begin{bmatrix} p_i & p_{i-1} & \cdots & p_{i-k} \end{bmatrix} \in \mathbf{R}^{3\times(k+1)},$$

$$\underline{t} = \begin{bmatrix} 1 & t-i & \cdots & (t-i)^k \end{bmatrix}^T \in \mathbf{R}^{(k+1)},$$

i is an integer that satisfies $i \leq t < i+1$ (i.e., the integer part of t), and $A_k \in \mathbf{R}^{(k+1)\times(k+1)}$ is a constant coefficient matrix. In particular, A_3 is determined from (3) as

$$A_3 = \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 3 & 3 & -3 \\ 4 & 0 & -6 & 3 \\ 1 & -3 & 3 & 1 \end{bmatrix}.$$

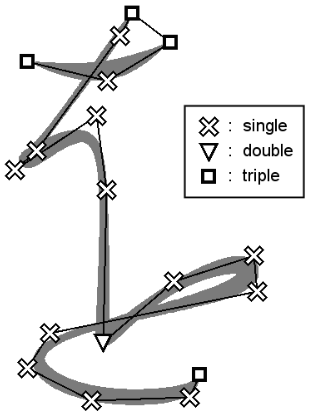


Fig. 3 Control polygon and resulting character.

2.2 Control Points and Polygon

We can design a desirable writing motion by specifying an appropriate set of control points:

$$p_{l-k+1},\ p_{l-k+2},\ \cdots,\ p_{m+k-1}.$$

Such a sequence of the control points then yields a polygonal line called "control polygon", and it represents a geometrical outline of the writing motion. Moreover, the control polygon M denoted by

$$M = p_{l-k+1}\ p_{l-k+2} \cdots p_{m+k-1}$$

can be regarded as a formal representation of the writing motion [5].

On the other hand, we are often interested in a writing motion that starts and terminates stationary, and such a motion can be realized by imposing the boundary conditions

$$\frac{d^j p(t_{l+1})}{dt^j} = \mathbf{0},$$
$$\frac{d^j p(t_{m+k})}{dt^j} = \mathbf{0}, \quad j = 1, \cdots, k-1. \qquad (4)$$

In the case of $k = 3$, (4) indicates that the velocity and the acceleration of $p(t)$ at the boundaries are zeros. In general, such a motion can be obtained by forcing the first, and similarly the last, k weighting vectors to be identical (i.e. by taking multiple control point with multiplicity k), namely

$$p_{l-k+1} = p_{l-k+2} = \cdots = p_l \qquad (5)$$

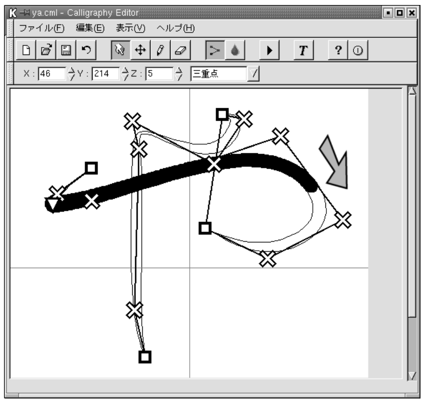
and

$$p_m = p_{m+1} = \cdots = p_{m+k-1}. \qquad (6)$$

In the following, we assume unless otherwise stated that cubic splines ($k = 3$) are used for planning writing motions, and also that the condition (4) i.e. (5) and (6) are specified. **Figure 3** shows an example of generated character 'wo' together with the corresponding control polygon projected onto the $O - XY$ plane,

Table 1 Development environment.

CPU	600 MHz Intel Pentium III
Architecture	IBM PC compatible
OS	Laser5 Linux 6.4
Graphics library	GTK+
XML library	libxml



X : 46 Y : 214 Z : 5 三重点

Fig. 4 Screen shot of the font editor.

where cross, triangle and square marks respectively denote single, double, and triple control points. It may be observed that single and double control points are arranged close to the character's body, and triple control points are placed at the starts and goals of each stroke.

3. Interactive Font Design System

As we have seen, writing motion is generated by specifying an appropriate set of weighting vectors, or equivalently a control polygon. Thus, any operations on dynamic fonts reduce to those on control polygons, and can be used for designing and editing fonts. Translation, scaling, rotation, and concatenation are examples of basic operations, and moreover more advanced operations such as smoothing (see Section 5) can be defined as well.

3.1 Font Editor

Based on such an idea, we developed an interactive design software, a font editor, and named "CE" (Calligraphy Editor). A development environment of the software is shown in **Table 1**, and an example of a screen shot is shown in **Fig. 4**. CE provides graphical and interactive user interfaces with various basic functions. They include addition or removal, displacement in XYZ direction, and changing multiplicities, of control points. Most of these operations are achieved as mouse operations. In terms of the characters, these operations respectively enable us to

Table 2 CML specification.

Name of tag	Attributes	Values	Description
`<control-polygon>`	`width`, `height`	$width, height > 0$	root node of document
`<point>`	`x`, `y`, `z` `m` (multiplicity) `w` (reliability)	$x, y, z \in \mathbf{R}$ $m \in \{1, 2, 3\}$ $w \in (0, 1]$	control point
`<ref>`	`file`	*file name*	external data reference
`<scale>`	`x`, `y`, `z`	$x, y, z > 0$	scaling operation
`<translate>`	`x`, `y`, `z`	$x, y, z \in \mathbf{R}$	translating operation
`<rotate>`	`phi`, `theta`, `psi`	$\phi, \theta, \psi \in [-\pi, \pi]$	rotating operation
`<concatenate>`	`kappa` (joint strength)	$\kappa \in [0, 1]$	concatenating operation
`<smooth>`	`lambda` (smoothing parameter)	$\lambda > 0$	smoothing operation

(F1) manipulate more precisely or straighten the relevant stroke,

(F2) adjust the curvature as well as the thickness, and

(F3) change the cursiveness or speed of the writing motion.

Computational amount for editing fonts is small and the computational time may be considered negligibly small in nowadays computers. This is due essentially to the simple form in (1) for generating the trajectory. For example, let us consider displacing the i-th control point by mouse dragging to re-design the cursiveness of certain stroke partially. Since the effect of adjusting the weight p_i for the unit motion $\mathbf{1}B_3(\alpha(t-t_i))$ remains only in the time interval $[t_i,\ t_{i+4}]$, trajectories outside of this interval need not be renewed. The resulting fonts are visualized almost instantaneously, and it enables us to design fonts easier.

Various operations on control polygons, both basic and advanced ones mentioned above, are also incorporated in CE by specifying appropriate parameters on pop-up dialogue box, or by some interactive operations. Thus a larger size of font for instance is obtained readily by specifying a scalar greater than one, which corresponds to enlarge the original control polygon.

In addition, the speed of writing motion can be changed by adjusting the time-scaling parameter α in (1) and the motion can be visualized. Such functions are also implemented in CE as ‘animation view’, and it visualizes the writing process of characters dynamically as if they are being written by human in real time.

3.2 Descriptive Language for Font Data

In order to record the control polygonal data edited by the software, we need to design a certain format suitable for the font development environment. The specific requirements here are that not only the font data are stored but various operations on fonts can be realized easily and efficiently. With these considerations, we propose a descriptive language for font data for use in CE, and named “CML” (Calligraphy Markup Language). CML is designed subject to the specification of XML. Using the hierarchical data structure of XML, the relationship between font and operations on font can be described as the relationship between the child node and the parent node in XML tree.

```
<?xml version="1.0"?>
<control-polygon canvas="500x500">
  <point x="15" y="217" z="0" m="3"/>
  <point x="-49" y="99" z="-48" m="1"/>
  <point x="-73" y="21" z="-12" m="2"/>
  <point x="-27" y="127" z="-5" m="1"/>
  <point x="3" y="-215" z="0" m="3"/>
</control-polygon>
```

Fig. 5 Example of primitive data.

```
<?xml version="1.0"?>
<control-polygon canvas="700x700">
  <scale x="1.5" y="1.6" z="1.5">
    <ref file="ma.cml"/>
  </scale>
</control-polygon>
```

Fig. 6 Example of an external reference with an operation.

Table 2 shows the specification of the language, where name of tags, attributes, and values with their domains are shown. **Figures 5** and **6** are description examples, showing how CML can be used in practice. Figure 5 describes the data of a character using the set of ordered `<point>` tags which corresponds to the control polygon. In Fig. 6, such a primitive form of data is reused and operated, where `<ref>` tag denotes an external reference to stored CML data specified in ‘`file`’ attributes, and `<scale>` $\cdots$ `</scale>` represent a scaling operation on the bracketed data, resulting in scaled charac-

Table 3 Statistics of cursive Kana font data.

Number of characters	48
Number of strokes	1.85 (average)
Number of distinct control points	13.4 (average)
Data size (bytes)	615 (average)
Standard output size (pixel)	500 × 500

ters. Such references and operations in Fig. 6 are readily processed, and the data is converted into the primitive form consisting merely of `<point>` tags (e.g., Fig. 5).

Such an interpretation of CML is also implemented in the software, and it is used for producing various transformed characters, moreover words and sentences.

4. Smooth Connection of Characters

Using the editor CE, we designed Japanese cursive font as summarized in **Table 3**. The font includes forty-eight basic Kana characters, and it is designed so as to resemble traditional brush-written style script based on a calligraphy textbook[16].

Every character in this set can be linked in arbitrary combinations to generate arbitrary words or sentences. It is obvious that the first step for generating successive characters is to translate each characters or control polygons in XY direction, since they are originally designed in some default position. This is a straightforward task, and thus our main concern here is how to connect the control polygons in a systematic fashion, and this process can be best described by example.

Figure 7 illustrates the process of generating the word *'tori'* from characters *'to'* and *'ri'*, where *'tori'* is generated by obtaining another CML code shown in **Fig. 8**. First, two control polygons corresponding to *'to'* and *'ri'* are translated in Y direction and then connected to produce one larger control polygon. Here, as the left figure of Fig. 7 shows, straightforward connection does not yield smooth cursive word. This is due to the fact that, although we now have one larger control polygon, all the initial and terminal points of the original control polygons have been designed as triple control points. When connecting the control polygons, we thus need to remove or reduce multiplicities of those control points. The function of connecting control polygons with such a care is realized as the operation represented by `<concatenate>` ··· `</concatenate>`. At present, we designed the operator so as to remove the last control

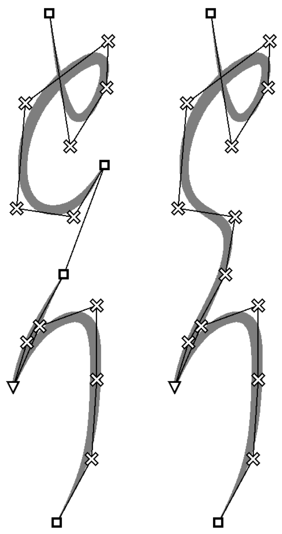


Fig. 7 Japanese cursive characters, *'to'*, *'ri'* and the word *'tori'*.

```
<?xml version="1.0"?>
<control-polygon canvas="500x1000">
  <concatenate>
    <translate y="250">
      <ref file="to.cml"/>
    </translate>
    <translate y="-250">
      <ref file="ri.cml"/>
    </translate>
  </concatenate>
</control-polygon>
```

Fig. 8 Data for generating successive characters.

point of the preceding control polygon and reduce the multiplicity of the first control point of the succeeding polygon from triple to single. Applying such an operation, we have continuous non-stationary writing motion yielding smooth successive string (See the right figure in Fig. 7).

It is noted that the locality property of the writing motion (1) again plays important roles for generating successive characters. The effect by linking two control polygons with the modified boundary control points is limited to only a local time interval, and the trajectory outside of the interval is left unchanged. The process of connecting characters is thus stable numerically without producing any oscillations as often observed in interpolation-based method.

5. Cursive Fonts and Sentences

In Japanese calligraphy, Kana characters are often written in omitted style to make concatenations smooth and continuous, and it is believed that the best way for learning techniques of writing such scripts is by copying ancestor's works over and over again. Although the omitted scripts have been generated by humans in some sensitive or artistic way, it may be useful for font designer if they could produce characteristic cursive characters in a systematic way. Here we propose a smoothing operation on fonts using the theory of optimal smoothing splines [8),9)]. The operation can also be represented as the operation on writing motion, or equivalently on control polygon.

5.1 Smoothing Operation

From a given motion $p(t)$ in (1) corresponding to a 'standard' writing motion, we consider to construct a family of motions $\mathcal{X}$ consisting of motions $x(t)$ of the same form as $p(t)$, namely

$$\mathcal{X} = \left\{ x(t) : x(t) = \sum_{i=l-k+1}^{m+k-1} \tau_i B_k(\alpha(t-t_i)) \right\},$$

where $\tau_i \in \mathbf{R}^3$. Equivalently, given a set of control points $\{p_{l-k+1}, \cdots, p_{m+k-1}\}$, we reconstruct other sets of control points $\{\tau_{l-k+1}, \cdots, \tau_{m+k-1}\}$ in such a way that it corresponds to the 'smoothed' writing motions. Note that, in this stage, there are no differences between a character and a sequence of characters since they can be treated in the same framework, namely as one control polygon. In other words, the smoothing operation to be defined below can be applied to both characters and character strings.

For simplicity, we construct motions $x(t)$ in three-dimensional space from one-dimensional motions smoothed for each of the three elements independently. Employing a notational abuse, namely with the understanding that $x(t)$, p_i, τ_i represent typical elements of the original vectors, such a smoothing operation is formulated as the following optimization problem.

$$\min_{x \in \mathcal{X}} \left(\lambda \int_{t_{l+1}}^{t_{m+k}} \{x''(t)\}^2 \, dt + \sum_{i=l-k+1}^{m+k-1} w_i (p_i - \tau_i)^2 \right). \tag{7}$$

Here λ is a positive constant scalar employed for adjusting smoothness property of $x(t)$, and $w_i \in (0,1]$ $(i = l-k+1, \cdots, m+k-1)$ is a parameter that represent the reliability of each i-th control point p_i. Note that analogous criteria are used in the theory of splines [8),9)], and edge detection problem in image processing [14)].

It can then be shown that Eq. (7) is rewritten simply as

$$\min_{\tau \in \mathbf{R}^N} \left(\lambda \alpha^3 \tau^T V \tau + (p-\tau)^T W (p-\tau) \right)$$

where $N = (m+k-1) - (l-k+1) + 1 = m - l + 2k - 1$,

$$\tau = [\tau_{l-k+1} \cdots \tau_{m+k-1}]^T \in \mathbf{R}^N,$$
$$p = [p_{l-k+1} \cdots p_{m+k-1}]^T \in \mathbf{R}^N,$$
$$W = \operatorname{diag}\{w_{l-k+1}, \cdots, w_{m+k-1}\} \in \mathbf{R}^{N \times N},$$

and $V \in \mathbf{R}^{N \times N}$ is a symmetric nonnegative-definite matrix with the ij-th element $v_{i,j}$ defined by

$$v_{i,j} = \int_{t_{l+1}}^{t_{m+k}} B_k''(t-i) B_k''(t-j) dt.$$

Using (2) and (3), the matrix V is obtained explicitly as

$$V = \bar{V} - (V_1 + V_2),$$

where the ij-th element $\bar{v}_{ij}$ of $\bar{V}$ is defined by

$$\bar{v}_{ij} = \begin{cases} 8/3 & i-j=0 \\ -3/2 & |i-j|=1 \\ 0 & |i-j|=2 \\ 1/6 & |i-j|=3 \\ 0 & \text{otherwise} \end{cases},$$

and

$$V_1 = \frac{1}{6} \left[\begin{array}{ccc|c} 14 & -6 & 0 & \\ -6 & 8 & -3 & 0_{3,N-3} \\ 0 & -3 & 2 & \\ \hline \multicolumn{3}{c|}{0_{N-3,3}} & 0_{N-3,N-3} \end{array} \right],$$

$$V_2 = \frac{1}{6} \left[\begin{array}{c|ccc} 0_{N-3,N-3} & \multicolumn{3}{c}{0_{N-3,3}} \\ \hline & 2 & -3 & 0 \\ 0_{3,N-3} & -3 & 8 & -6 \\ & 0 & -6 & 14 \end{array} \right].$$

Then, the solution to this minimization problem is uniquely given by

$$\tau = (\lambda \alpha^3 V + W)^{-1} W p.$$

Actually we solve the system of linear algebraic equations

$$(\lambda \alpha^3 V + W)\tau = Wp,$$

which is stable numerically since the coefficient matrix is positive-definite and banded matrix.

This process is repeated for each of the three elements of $p(t)$, all with the same values of design parameters λ and w's.

5.2 Examples of Smoothing

Here we present some examples showing how such a smoothing operation is utilized for changing the style of characters. **Figure 9** illustrates the word *'aki'*. Shown in Fig. 9

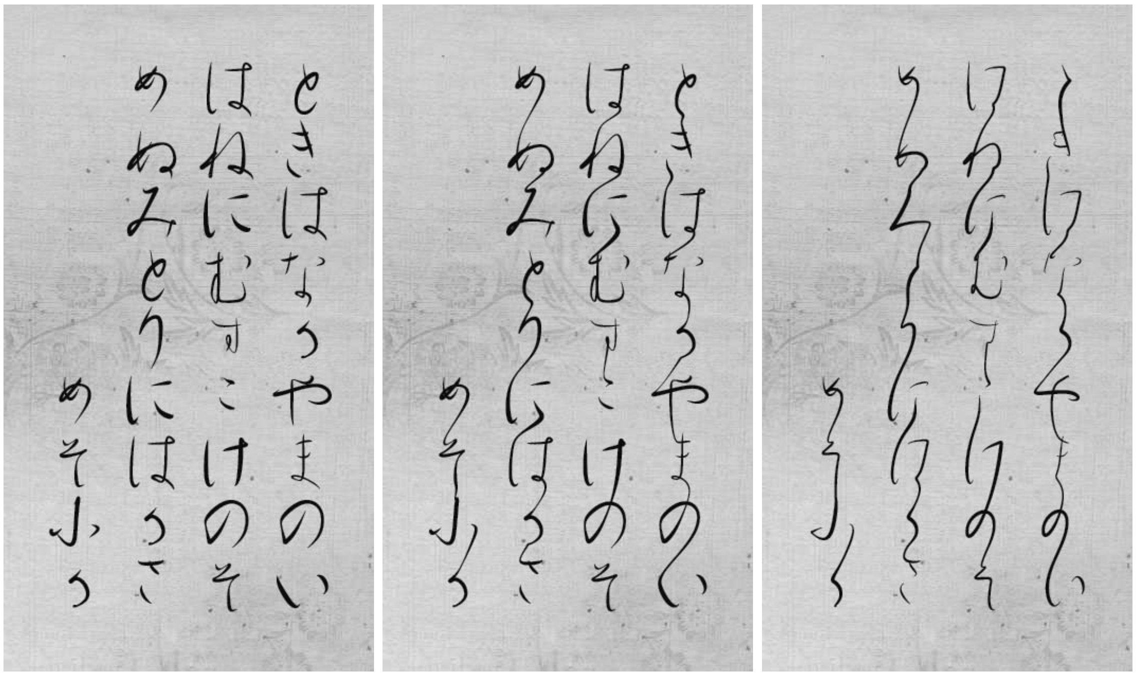


Fig. 10 Three Japanese sentences generated by translation, concatenation, and smoothing operations (from left).

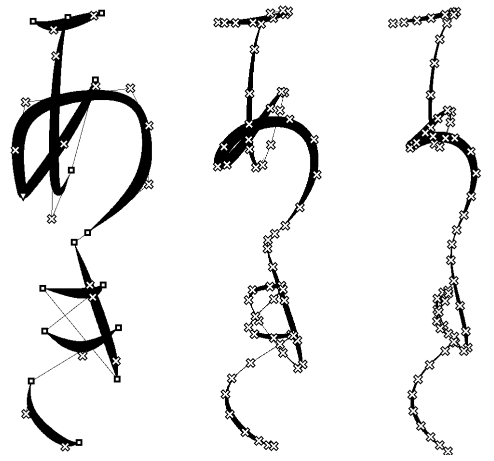


Fig. 9 Original word and two reconstructed words by smoothing operation with $\lambda = 2.0$ and 8.0 (from left).

are, from the left, standard word with original control points $\{p_{l-k+1}, \cdots, p_{m+k-1}\}$, and two smoothed words together with re-constructed control points $\{\tau_{l-k+1}, \cdots, \tau_{m+k-1}\}$, obtained by specifying parameters $\lambda = 2.0$ and 8.0 both for $W = I$. It may be observed that, as λ increases or as we put more weight on the smoothness term in (7), more cursive characters are produced.

Figure 10 shows Japanese poem '*tanka*' written in four lines. The left figure shows the sentence generated by only translation operation, the middle one is generated by concatenation, and the right figure shows transformed sentence obtained by applying the smoothing operation with $\lambda = 1.2$ and $W = I$. It is noted that the smoothness of each character itself, as well as the smoothness between the characters are increased. We observe that such naturally smoothed sentences as if brush-written by humans can be produced by employing the simple, mathematical operation, i.e. smoothing. Obviously, larger values of the smoothing parameter λ result in sentences in more deformed style.

For the practical use of the proposed scheme, it is useful to store output sentences in some graphic formats, such as SVG (Scalable Vector Graphics) [15]. Using SVG, the authors have thus developed an experimental system on a web site, and named 'Calligraphy on Demand'. In the web site☆, user can input any *tanka* regardless of his original or well-known ones. Then by specifying the continuousness, smoothness, color of ink, and background paper, the user can create original writing works as is shown in Fig. 10 interactively.

☆ Calligraphy on Demand
URL: http://kanolab.j.dendai.ac.jp/˜hiro/cod/

6. Concluding Remarks

Based on the generation method of dynamic fonts, we developed an interactive font design software together with a descriptive language for control polygonal data. Using the editor, we designed brush-written Kana font and we have shown that the characters in the font can be concatenated easily by applying the operations on respective control polygons. We also developed smoothing operation on writing motion, or control polygon representing either a character or a sequence of characters, by introducing the theory of optimal smoothing splines. Some examples have shown that the operation is effectively used for generating naturally smoothed brush-written sentences which resembles artworks often seen in Japanese calligraphy.

The novelty of the method lies in the fact that whole of the process, namely from designing each characters up to generating cursive sentences, can be treated in a unified framework as well as in a systematic way. More specifically, in the present method, there are no essential differences in treating a character, a word, a sentence and sentences, since they are all represented in the same framework, namely by a control polygon either small or large. As the result, any operators developed for a control polygon such as smoothing can be applied irrespective whether it represents a character, a word, and a sentence. Such a systematic treatment could not be possible in the conventional methods. Although our attension has been focussed on Kana characters, we believe that the scheme can be applied to other types of characters as well. In particular, Chinese characters certainly seem to be suited to the present scheme, since each character in general is constituted from several basic elements that may be modeled by control polygons.

Further studies are needed in the following points. One is to develop a proper method for evaluating generated characters. Kana invented in Japan in 12th century by nobles has the long history. Various techniques that have been practiced traditionally in Kana calligraphy, such as arrangement, concatenation, and omission of characters are due in most part to a sense of beauty[16),17)]. It is thus desirable if we could analyze such a sense of beauty mathematically, and devise some aesthetic evaluation method of output sentences. The other important point is to develop a more realistic generation model mimicking writing device and process in calligraphy. For example, we may consider a virtual writing brush device which changes its shape dynamically[18)], or writing motions generated by virtual robotic arm holding the device.

Acknowledgments The authors would like to thank Professor Clyde F. Martin of Texas Tech University for stimulating discussions on smoothing splines. Also, special thanks are extended to Mrs. Shijyun Sugiyama, Director of Modern Calligraphy Research Institute, for valuable advices for improving the aesthetic quality of output sentences.

References

1) Uehara, T.: Current Technology and Problems in Computer Font (in Japanese), *Trans. Inf. Process. Soc. Jpn.*, Vol.31, No.11, pp.1570–1580 (1990).

2) Sakamoto, M. and Takagi, M.: Computerized Generation of High-Quality Fonts of the Japanese Alphabets (in Japanese), *Trans. Inst. Electrons. Inf. Commun. Engrs.*, D-II, Vol.68, No.4, pp.702–709 (1985).

3) Tokura, T., Suzuki, T., Nakamura, H., Makino, Y. and Takakura, S.: Computerized Generation Systems of Brush-Written Japanese Characters with Inter-character Connections (in Japanese), *Trans. Inf. Process. Soc. Jpn.*, Vol.29, No.1, pp.20–28 (1988).

4) Zhang, X., Ji, H., Sanada, H. and Tezuka, Y.: Forming Brush-Written HIRAGANA with the Capability of Providing Versatile Stroke-Connecting Flows (in Japanese), *Trans. Inst. Electrons. Inf. Commun. Engrs.*, Vol.76-D-II, No.9, pp.1868–1877 (1993).

5) Takayama, K. and Kano, H.: A New Approach to Synthesizing Free Motions of Robotic Manipulators Based on a Concept of Unit Motions, *IEEE Trans.SMC.*, Vol.25, No.3, pp.453–463 (1995).

6) Takayama, K., Kano, H., et al.: Dynamic Font: A New Representational Technology, *FUJITSU Scientific and Technical Journal*, Vol.32, No.2, pp.192–202 (1996).

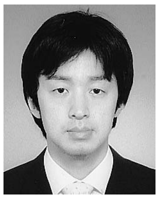
7) de Boor, C.: *A Practical Guide to Splines*, Springer-Verlag, New York (1978).

8) Kano, H., Egerstedt, M., Nakata, H. and Martin, C. F.: B-splines and Control Theory, *Applied Mathematics and Computation*, accepted for publication.

9) Kano, H., Nakata, H. and Martin, C. F.: Optimal Design of Curves by B-Splines, *Proc. Int. Symp. on Stochastic Systems Theory*, pp.164–

169, Ashikaga (2001).
10) Nakata, H. and Kano, H.: Interactive Design Method for Dynamic Fonts (in Japanese), *Proc. 2001 IEICE General Conf.*, Vol.2, pp.190, Kusatsu (2001).
11) Nakata, H. and Kano, H.: Generation of Japanese Cursive Fonts Using B-Spline Functions (in Japanese), *Proc. 63th Nat. Conv. IPSJ*, Vol.2, pp.205–206, Yamaguchi (2001).
12) Nakata, H. and Kano, H.: Generation of Japanese Cursive Fonts Using B-Spline Functions, *Proc. IASTED Int. Conf. Intelligent Systems and Control*, pp.475–479, Clearwater, Florida (2001).
13) Nakata, H. and Kano, H.: Generation of Japanese Cursive Sentences Using Optimal Smoothing Splines (in Japanese), *Proc. 64th Nat. Conv. IPSJ*, Vol.4, pp.127–128, Saitama (2002).
14) Faugeras, O.: *Three-Dimensional Computer Vision — A Geometric Viewpoint*, The MIT Press (1996).
15) W3C, Scalable Vector Graphics (SVG). http://www.w3c.org/Graphics/SVG/
16) Kuwata, S.: *The Beauty of Kana* (in Japanese), Shodan Koron Publishing (1985).
17) Enokura, K.: *Kana Calligraphy* (in Japanese), Japan Broadcast Publishing (1998).
18) Saito, S. and Nakajima, M.: 3D Physics-based Brush Model for Interactive Painting (in Japanese), *Trans.Inf.Process. Soc.Jpn.*, Vol.41, No.3, pp.608–615 (2000).

(Received March 19, 2002)
(Accepted October 7, 2002)



Hiroaki Nakata was born in Yokohama, Japan, in 1976. He received his B.S. and M.S. degrees in Information Sciences from Tokyo Denki University, Saitama, Japan, in 2000 and 2002, respectively. His interests are in font generation and XML applications. He is a member of IPSJ.

Hiroyuki Kano received the B.E. and M.E. degrees in Mechanical Engineering from Kyoto Institute of Technology, Kyoto, Japan, in 1970 and 1972, respectively, and the Ph.D. degree in Electrical Engineering-System Science in 1976 from Polytechnic Institute of New York (now Polytechnic University), New York, NY. From 1976 to 1992, he was with the International Institute for Advanced Study of Social Information Sciences, Fujitsu Ltd., Japan. Subsequently, in 1992, he joined Tokyo Denki University, Saitama, Japan, as Professor in the Department of Information Sciences. His interests include vision-based robots, signal processings, and control theory and applications. He is a member of the SICE, ISCIE, RSJ, JSIAM, and IEEE.