Automatic Generation of Typographic Font from a Small Font Subset

Tomo Miyazaki, Tatsunori Tsuchiya, Yoshihiro Sugaya, Shinichiro Omachi, Masakazu Iwamura, Seiichi Uchida, and Koichi Kise,

***Abstract*—This paper addresses the automatic generation of a typographic font from a subset of characters. Specifically, we use a subset of a typographic font to extrapolate additional characters. Consequently, we obtain a complete font containing a number of characters sufficient for daily use. The automated generation of Japanese fonts is in high demand because a Japanese font requires over 1,000 characters. Unfortunately, professional typographers create most fonts, resulting in significant financial and time investments for font generation. The proposed method can be a great aid for font creation because designers do not need to create the majority of the characters for a new font. The proposed method uses strokes from given samples for font generation. The strokes, from which we construct characters, are extracted by exploiting a character skeleton dataset. This study makes three main contributions: a novel method of extracting strokes from characters, which is applicable to both standard fonts and their variations; a fully automated approach for constructing characters; and a selection method for sample characters. We demonstrate our proposed method by generating 2,965 characters in 47 fonts. Objective and subjective evaluations verify that the generated characters are similar to handmade characters.**

***Index Terms*—Font generation, Active shape model, Computer aided manufacturing, Kanji character.**

I. Introduction

Automatic generation of Japanese font is important for lowering the costs associated with creating kanji[1] characters. All of the characters in a font need to be designed in a particular style and size, a process generally performed by humans. Moreover, font creation is professional and time-consuming work.

The cost problem is especially crucial in languages that contain a very large number of characters, such as Chinese, Korean, and Japanese. Most commercial fonts contain at least 1,006 kanji characters, which are defined as those in daily use by the Japanese Industrial Standards Committee[2]. Chinese and Korean each use more than 6,000 characters. These numbers are far greater than those of other major languages, e.g., 26 characters in English, 27 in Spanish, 49 in Hindi, 28 in Arabic, and 33 in Russian. Thus, it takes a few years even for a professional designer to create a font.

Previous work on character generation is shown in Table I. We categorized previous studies by their inputs and outputs into three problem settings. The first problem setting is to generate a font using parameters for handwritten [1]–[6] or typographic fonts [7]–[12]. The second problem setting is to generate a font by blending several complete fonts[3] for handwritten [13]–[15] or typographic fonts [16]–[19]. The third problem setting is to generate a font by extrapolating characters given a subset of handwritten [20]–[27] or typographic fonts [28]. Each problem setting has been an important research topic.

In this study, we address the problem of generating a typographic font from a given subset of characters. From a subset of a typographic font that includes only a small number of characters, we generate missing characters to obtain a complete typographic font. Although the proposed method requires a subset of a particular typographic font, the burden of creating a subset of a font is much less than creating an entire font. This benefit is highlighted in languages containing many characters, such as Chinese and Japanese. The proposed method uses sample characters to extract strokes and constructs characters by deploying them. Fig. 1 illustrates an overview of the proposed method, which begins with stroke extraction from sample characters. Strokes are extracted using the skeletons of the samples. We take the skeletons of a target character from the skeleton dataset and transform it into the structure of the target font. Finally, we select some strokes and deploy them to the target skeletons.

The proposed method accepts samples in an image format. For practicality, the ideal font generation method must accept characters in an image format and require only a small subset of the font as input. First, we consider the image format. For the purpose of character recognition, character generation from samples in an image format is far more useful than from those in a vector format. Image formats are more convenient for sample collection. For example, when users encounter text in an unknown font, it is difficult to find samples in a vector format, whereas an image is available immediately upon taking a photo of the text. Likewise, sample collection is time-consuming work and could be an obstacle for the popularization of a font generation method if too many samples are needed.

Our automatic generation framework makes three main contributions, each of which is briefly introduced below and described in detail in subsequent sections.

The first contribution of this study is a stroke extraction method. We propose a method that extracts character strokes by applying a novel active contour model that determines the boundaries of strokes to extract. Some strokes are damaged because of the complexity of characters; therefore, we incorporate a restoration process for fixing these damaged strokes.

The second contribution of this study is a method for the automatic construction of characters in two phases: modifying skeletons in the target font and deploying strokes to them. The proposed method extracts the transformation parameters. With these parameters, we can unify the skeletons in size and geometry. Because our method transforms skeletons from a standard font (such as Mincho) to fit the target font, it is unnecessary to build skeleton datasets for a large number of fonts. This approach helps us with variety and scalability. We can generate as many characters as the skeletons in the dataset will allow (210,000 characters are available at this time). For automation, we define an energy function to deploy a stroke so that an appropriate one can be added to the skeleton.

The third contribution of this study is a sample selection method. We address which characters are suitable for the proposed method and how we can select them based on the energy function used in character generation. Finding samples is combinatorial problem, and it is difficult to obtain a solution analytically. Therefore, we apply a

[1]Kanji are Chinese characters adapted for the Japanese language.

[2]http://www.jisc.go.jp

[3]A complete font contains a sufficient number of characters for daily use, e.g., 26 for English and over a thousand for Japanese.

TABLE I
CATEGORIES OF RELATED WORK

Input \ Output	Handwriting font	Typography font
Parameters (Parameter-based methods)	[1] [2] [3] [4] [5] [6] $\{\theta_1,\cdots\}\rightarrow\{A,\cdots,Z\}$	[7] [8] [9] [10] [11] [12] $\{\theta_1,\cdots\}\rightarrow\{A,\cdots,Z\}$
Complete fonts (Font blending methods)	[13] [14] [15] $\{A,\cdots,Z\}$ ⋮ $\{a,\cdots,z\}\rightarrow\{A,\cdots,Z\}$	[16] [17] [18] [19] $\{A,\cdots,Z\}$ ⋮ $\{a,\cdots,z\}\rightarrow\{A,\cdots,Z\}$
An incomplete font (Character extrapolation methods)	[20] [21] [22] [23] [24] [25] [26] [27] $\{A,B\}\rightarrow\{A,\cdots,Z\}$	[28] [This paper] $\{A,B\}\rightarrow\{A,\cdots,Z\}$

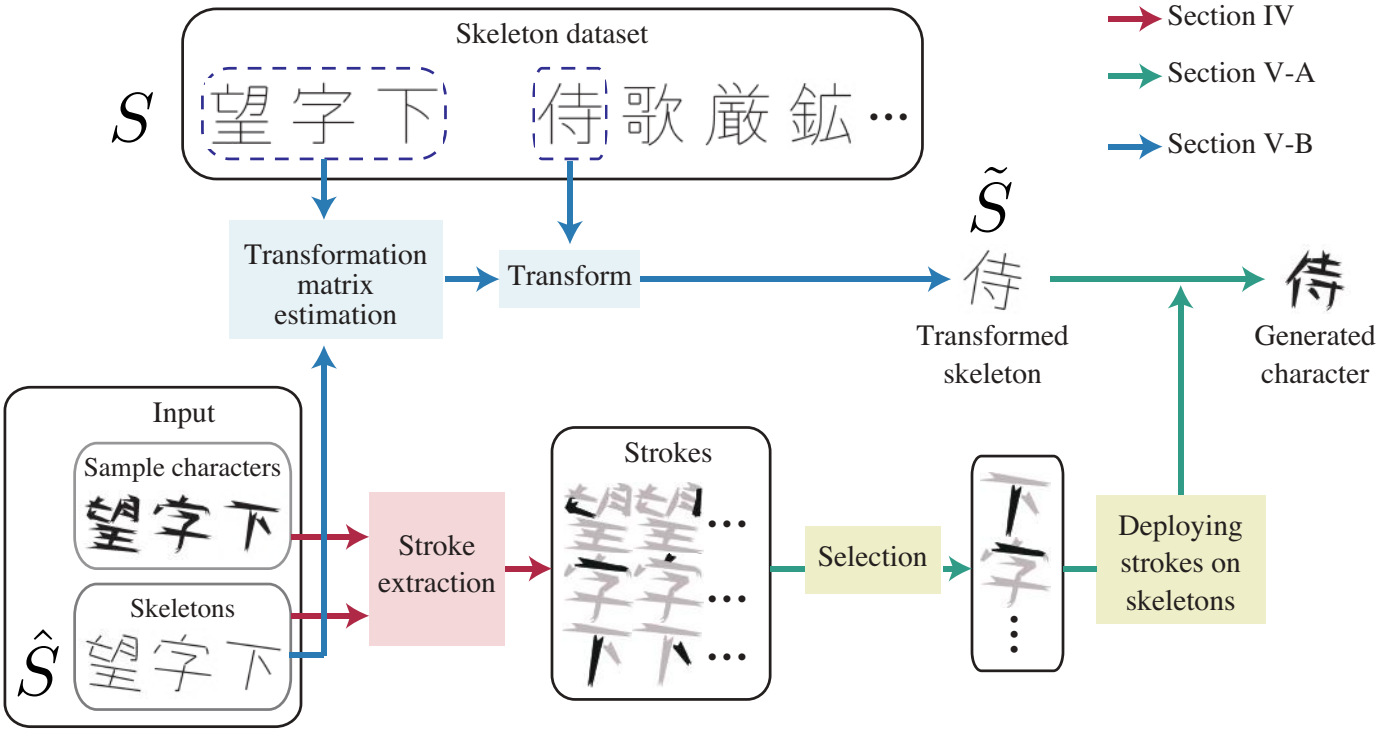


Fig. 1. Overview of the proposed method.

genetic algorithm to find an approximate solution.

II. Related work

In this section we review the relevant work illustrated in Table I.

Parameter-based methods: The methods in this category use parameters acquired by analyzing fonts. One of the most famous methods is Metafont [7], which use the shapes of pen tips and pen-movement paths as parameters for character generation. [8] was inspired by Metafont. Further, [9], [10] analyzed characters in the Times font family to extract parameters. Character properties, such as thickness and aspect ratio, are changed using these parameters. A method for generating Japanese characters was addressed by Tanaka *et al.* [11], who created a database of skeletons of Japanese characters, including kanji. The characters were generated by placing thick lines in the skeleton data. A famous noncommercial font[4] was created by applying this method. Kamichi extended the skeleton database in [12].

There have been attempts to analyzing handwritten characters. Bayoudh *et al.* used Freeman chain-code to analyze alphabetical characters [1]. Djioua and Plamondon used a Sigma lognormal model supported by the kinematic theory [2]. An interactive system was developed so that the user can easily fit a Sigma-lognormal model to alphabetical characters. Wada *et al.* extracted the trajectories of alphabetical characters and replaced them using a genetic algorithm [3]. Zheng and Doermann adopted a thin plate spline to model an alphabetical character and generated a new alphabetical character by calculating the intermediate of the two [4]. Handwriting models for robot arms were developed in [5], [6].

[4]https://www.tanaka.ecc.u-tokyo.ac.jp/ktanaka/Font/

Parameter-based methods generate clean characters. However, they are only applicable to particular fonts that are analyzed by humans. For instance, the method in [11] accepts only two fonts: Mincho and Gothic. This drawback comes from the method's requirement for precise analysis. In addition, the methods cannot be applied to Japanese fonts, since over a thousand of characters need to be parameterized. Moreover, special equipment is required to accurately parameterize alphabetical characters, such as an interactive system or device for acquiring trajectories. Unlike these methods, the method proposed in this study generates characters without requiring precise analysis. Thus, only a few character sample images are required. As we discussed in the previous section, preparing sample images is a simple task.

Font blending methods: Methods in this category receive several complete fonts and generate a font by blending them. Xu *et al.* generated Chinese calligraphy characters using a weighted blend of strokes in different styles [13]. They decomposed samples into radicals and single strokes based on rules defined by expert knowledge. An improved method [14] considers the spatial relationship of strokes. Choi *et al.* generated handwriting Hangul characters using a Bayesian network trained with a given font [15].

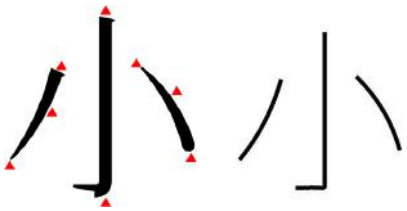
Suveeranont and Igarashi addressed a generation of alphabetical characters for typographic fonts [16]. They generated characters by blending predefined characters from miscellaneous complete fonts. This method is based on a vector format, in contrast with the proposed method, which accepts an image format. Campbell and Kautz learned a manifold of standard fonts of alphabetical characters [17]. Locations on the manifold represent a new font. Feng *et al.* used a wavelet transform to blend two fonts [18]. Tenenbaum and Freeman used complete fonts as references to form characters from the generation results [19].

The methods in this approach require human supervision to generate a desired font; since they automatically blend fonts, the result is not always the desired font. In order to avoid this problem, blending weights [13] or an interactive system [16] are implemented. However, sweeping the weights and using the system require human supervision.

In contrast, the proposed method generates a typographic font with little human supervision. Instead of blending fonts, we construct characters with strokes extracted from sample characters in the desired font. Therefore, the proposed method can directly provide the user with the desired font.

Character extrapolation methods: The aim of methods in this category is to extrapolate characters not included in a given subset. Attempts to complete this process on handwritten fonts can be found in [20]–[27]. Lin *et al.* generated characters with components extracted from given characters [20] using an annotated font in which the positions and sizes of components were labeled. The extraction of the components was performed on electronic devices so that characters can be easily decomposed. Zong and Zhu developed a character generation method using machine learning [21]. They decomposed the given characters into components by analyzing the orientation of strokes. Components were assigned to a reference font with a similarity function trained by a semi-supervised algorithm. Wang *et al.* focused on the spatial relationships of the character components for decomposition and generation [22]. An active contour model was used for decomposition [23].

Character component decomposition is a crucial technique for the methods in this category. However, most use naive decomposition that rely on spatial relationships [22]–[26] or special devices [20], [27]. It is difficult to extract components when they are connected or decorated. To the best of our knowledge, the method in [28] is the only method that is applicable to characters in fonts with decorations. Saito *et al.* applied a patch transform [29] to samples and generated alphabetical characters in wide range of fonts [28]. However, the generated results did not meet the criteria for practical use. In this paper, we propose an adaptive active contour model for component extraction. With the proposed method, we can obtain natural character strokes even in decorated characters.



(a) (b)

Fig. 2. Visualization of the GlyphWiki data and our dataset. (a) Data from GlyphWiki drawn in Mincho. The red triangles represent the control points. The line types of the left and right strokes are "curve" and "straight" in the middle. All start and end shapes are different. (b) The skeletons in our dataset. The control points, line type, start shape, and end shape are attributes.

III. Skeleton Dataset

First, we address the character skeleton dataset used in this study. We created the dataset based on GlyphWiki [30], which contains data from more than 210,000 of characters[5]. GlyphWiki utilizes a wiki format, allowing anybody to contribute to and maintain the database. Each stroke of a character is stored in KAGE format [12] which uses four attributes: control points, line type, start shape, and end shape. There are six line types, seven start shapes, and 15 end shapes. The number of control points depends on line type and is at most four. The control points and line types lead to a rough stroke, then the start and end shapes provide details of a stroke. Fig. 2 (a) illustrates a character in KAGE format in Mincho.

We created skeletons of the strokes in a character using GlyphWiki data. Fig. 2(b) illustrates the skeletons of a character in our dataset. We draw a line of a skeleton from the first control point to the last control point. An additional line of a skeleton is drawn according to start shape and end shape. For example, the end shape is drawn in the middle stroke in Fig. 2(b). We draw a line according to the line type and the number of control points. A straight line is drawn when the line type is straight and two control points are given. A B-spline curve is drawn when the line type is curved and three control points are given. A straight line and B-spline curve are drawn when the line type is vertical and four control points are given. We include the attributes with each skeleton. We extract points on a skeleton by regular sampling from the start point to the end point. Therefore, a skeleton is composed of a set of sampling points. By using sampling points, we define a skeleton S as

$$S = \begin{bmatrix} \cdots & x_i & \cdots \\ \cdots & y_i & \cdots \\ \cdots & 1 & \cdots \end{bmatrix}, \tag{1}$$

where x_i and y_i represent the x- and y-coordinates of ith sampling point, respectively. We adopted homogeneous coordinates. $P = [x, y, 1]^{\mathrm{T}}$ denotes a vector that represents a sampling point. Then, $S = [\cdots, P_i, \cdots]$. We denote a skeleton in our dataset by S, a

[5]It includes characters created by users that are not used publicly.

Fig. 3. Stroke extraction. The black pixels represent the extracted strokes.

(a) (b)

Fig. 4. Skeleton adjustment GUI. (a) Screenshot of the GUI initiating an adjustment. Note that the skeleton is not matched to the sample at this point because the skeletons are in Mincho. (b) Adjustment result.

skeleton in the samples by $\hat{S}$, and a transformed skeleton[6] by $\tilde{S}$ (See Fig. 1).

IV. Stroke extraction

When the proposed method receives samples as an input, we extract the strokes of the samples. Fig. 3 shows examples of extracted strokes.

A. Adjusting skeletons to samples

The proposed method maximally exploits the skeletons of sample characters, however, the skeletons of samples are not given. The skeletons can be easily extracted if strokes are separated; however, strokes often overlap. Quality skeleton extraction is essential for the proposed method. Therefore, inspired by accurate segmentation applications, such as Grabcut [31], we adopt user interaction to adjust skeletons in our dataset to the samples so that accurate skeletons of the samples can be obtained.

We developed a graphical user interface (GUI) for adjusting skeletons, as illustrated in Fig. 4. Skeletons are displayed with control points on the sample. The operator adjusts a skeleton by dragging the control points and the GUI shows these changes in real time.

In order to assist the operator, we implemented three automation techniques: scale adjustment, rotation adjustment, and cooperative move. Firstly, a scale of skeletons is adjusted to fit the sample. We calculate a scaling factor from rectangles around the skeleton and sample. The second technique is rotation adjustment. We extract points from the skeleton by sampling. Likewise, we extract points from the medial axis of the sample, which is obtained by a morphological operation. With these points, we use iterative closest point matching [32] to calculate the rotation matrix. Thirdly, control points move cooperatively when the start point is moved by hand, e.g., if the start point moves up, the other control points also move up. The scale and rotation adjustments are performed only once, before manual adjustment.

We emphasize that skeleton adjustment is not difficult and can be completed in minutes. It is unnecessary to adjust all of the skeletons in the dataset. According to our experimental results, approximately 15 samples are sufficient for the proposed method. Therefore, skeleton adjustment is significantly less work than creating thousands of characters by hand.

B. Skeleton relation assignment

We determine the connectivity of the skeletons in the samples. We define relations between two skeletons as follows:

- continuous: the start or end of a skeleton is connected to the start or end of another.
- connecting: the start or end of a skeleton is connected to the body of another.
- connected: the body of a skeleton is connected to the start or end of another.
- crossing: the skeleton crosses over another.
- isolated: the skeleton is isolated from others.

The body is the part of a skeleton without start or end parts. Examples of each relation are illustrated in Fig. 5.

Two skeletons are in contact when the distance between them is small. Therefore, we measure the distance d between S and S' as

$$d = \begin{cases} \min\limits_{i,j} \|Q_{i,j} - P_i\|_2 + \|Q_{i,j} - P'_j\|_2 & \text{if } Q \text{ exists,} \\ \infty & \text{otherwise.} \end{cases} \tag{2}$$

$Q_{i,j}$ is the point at which two lines cross, which are vertical to S and S' and through P_i and P'_j, respectively. Fig. 6 illustrates an example of Q. Using the $\bar{i}$ and $\bar{j}$ that minimize d, we can determine which part of S is in contact with S', e.g., the start of S is in contact with the part of S' when $\bar{i}$ is small and $\bar{j}$ is large. We use three functions to indicate contact: $\mathbb{I}_s$ for the start, $\mathbb{I}_e$ for the end, and $\mathbb{I}_b$ for the body. For n points of a skeleton, the functions calculate true or false, as follows. $\mathbb{I}_s(\bar{i}) =$ True if $\frac{\bar{i}}{n} < .05$, otherwise False. $\mathbb{I}_e(\bar{i}) =$ True if $\frac{\bar{i}}{n} > .95$, otherwise False. $\mathbb{I}_b(\bar{i}) =$ True if $.05 \leq \frac{\bar{i}}{n} \leq .95$, otherwise False. The parameters $.05$ and $.95$ were determined experimentally. We assign a relation R to S against S' as follows:

$$R = \begin{cases} \text{continuous} & \text{if } d < 2(\tau + \tau') \\ & \quad \wedge (\mathbb{I}_s(\bar{i}) \vee \mathbb{I}_e(\bar{i})) \\ & \quad \wedge (\mathbb{I}_s(\bar{j}) \vee \mathbb{I}_e(\bar{j}))\,, \\ \text{connecting} & \text{if } d < 2(\tau + \tau') \\ & \quad \wedge (\mathbb{I}_s(\bar{i}) \vee \mathbb{I}_e(\bar{i})) \wedge \mathbb{I}_b(\bar{j}), \\ \text{connected} & \text{if } d < 2(\tau + \tau') \\ & \quad \wedge \mathbb{I}_b(\bar{i}) \wedge (\mathbb{I}_s(\bar{j}) \vee \mathbb{I}_e(\bar{j}))\,, \\ \text{crossing} & \text{if } d < 2(\tau + \tau') \\ & \quad \wedge \mathbb{I}_b(\bar{i}) \wedge \mathbb{I}_b(\bar{j}), \\ \text{isolated} & \text{otherwise,} \end{cases} \tag{3}$$

where τ represents a thickness of a stroke in a sample. We can draw a character image with τ. Fig. 7 illustrates the skeletons and created character with various values of τ. We empirically determined τ by creating character images with various τ and choosing the τ that minimizes the hamming distance between the created and sample character images.

We reconsider the relations obtained above. Since the relations are based on the skeletons, it is necessary to verify that they are consistent with the sample images. The relations may occasionally be different from the samples because of human error during adjustment and the

[6]We described the details of the transformed skeletion in Section V-A.

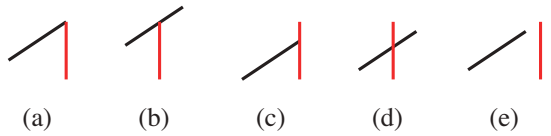


Fig. 5. Skeleton relations of the red skeletons to the black skeletons: (a) continuous, (b) connecting, (c) connected, (d) crossing, and (e) isolated.

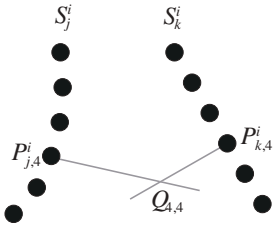


Fig. 6. Measurement of the distance between two skeletons. The sampling points are drawn with black circles.

parameters. The parameter $2(\tau+\tau')$ in Eq. (3) is set to a severe value in order to detect all stroke contacts without omission. Consequently, an incorrect relation may be assigned.

The procedures for reconsideration are as follows. We classify all pixels of the sample to the nearest skeleton within the Euclidean distance. Fig. 8 (a) illustrates the segmentation results. Subsequently, we choose a connected component from the sample using these results. Then, we determine the number of connected components obtained. If the relation is isolated, there must be two connected components. Likewise, if the relation is connecting, connected, or crossing, there must be only one connected component. This check complements Eq. (3).

C. Adaptive active contour model

We propose an adaptive active contour model (AACM) for character stroke extraction. The AACM is able to distinguish even strokes that have contact with others, determining the boundary of each stroke. Inspired by Snake [33], we seek the boundary by optimizing a spline curve. In addition, we incorporate constraints and adaptive energy modification into the AACM so that strokes can be extracted from complex characters.

The AACM is defined as a spline curve that minimizes the energy function as

$$E_{AACM} = E_{int} + E_{img}, \tag{4}$$

where E_{int} and E_{img} take into account the smoothness of the curves and the fit to objects, respectively. E_{int} follows [33]. E_{img} must be determined for each task. If we set E_{img} to $-|\triangle I|$, where $\triangle I$ represents the gradient image of I, E_{img} pulls the AACMs to the edges. For our purposes, it is unnecessary for a boundary to be very close to its stroke, and instead, may be rather relaxed. If the boundary is too close, it may cross the stroke. Consequently, the extracted strokes are damaged. Therefore, in this paper, we define E_{img} as

$$E_{img} = G_v * I_{SMP} + I_{SK}, \tag{5}$$

where G_v is a Gaussian kernel with variance v, and I_{SMP} and I_{SK} are the grayscale images[7] of the sample and its skeletons, respectively. E_{img} has a high energy[8] around the strokes, thus, the boundary remains separated from them. We use the boundaries of the segmentation results as the initial AACMs. We adaptively modify E_{img} and set some points that the AACMs must pass through.

In the case where the relation of the target stroke is isolated, which is the simplest case, we directly apply an AACM. Fig. 9 illustrates the results of the extraction of an isolated stroke.

In the case where the relation of the target stroke is connecting or continuous, we force an AACM to go through the point at which the strokes are in contact. Then, we decrease E_{img} by $1/\beta_1$ within the range of the length $\beta_2\tau$ from the point, except for I_{SK}. This decrease operation encourages the AACM to pass through the stroke intersection. β_1 and β_2 are constant values that were determined empirically. With the modified E_{img} and the constraints, we minimize the AACM and extract the target. Fig. 10 demonstrates the extraction for this case. The AACM naturally passes through the intersection due to the decreased energy.

Where the relation of the target stroke is crossing or connected, we decrease E_{img} by $1/\beta_1$ within the range of $\beta_2\tau_{others}$ from the other strokes. Then, we minimize the AACM with the modified E_{img} and extract the target along the minimized AACM.

D. Stroke restoration

We reshape extracted strokes, which often have defects at points where contact occurs; see Fig. 11(a). An intuitive method for restoration is the removal of contours around defects and the connection with cubic Bézier curves. A cubic Bézier curve is drawn with four points (P_1, P_2, P_3, P_4). The curve starts from P_1 and ends at P_4. P_2 and P_3 serve as control points that provide a direction to the curve.

We apply cubic Bézier curves to a stroke using POTRACE [34], generating a number of curves that represent the short parts of the contour. Then, we remove the curves within a distance of τ from the extracted stroke; see Fig. 11(c). We create a cubic Bézier curve between two curves so that the contour can be continuous; see

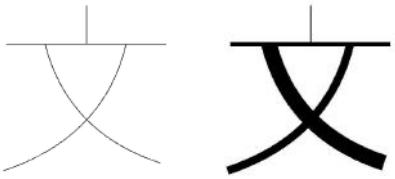
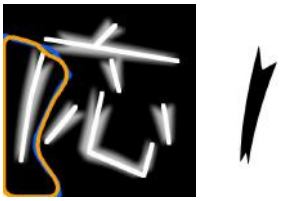


Fig. 7. (a) Skeletons and (b) created characters with $\tau = 1, 10, 20, 40$.

Fig. 8. Relation check. (a) Skeleton of segmentation results. (b) Sample of the segmentation results. Focusing on the two segments indicated by the red circle, the relation is isolated in (a), but there is only one connected component in (b).

[7]Pixel values are 0 if they are background, and 255 otherwise.

[8]The pixel value represents the energy: 0 is the lowest and 255 is the highest.



(a) (b)

Fig. 9. Extraction of an isolated stroke. (a) Initial AACM (blue) and minimized AACM (orange) on E_{img}. (b) Extracted stroke.

(a) (b)

Fig. 10. Stroke extraction for connecting or continuous stroke relations. (a) Initial AACM (blue) and minimized AACM (orange) on E_{img}. (b) Extracted stroke.

Fig. 11(d). We set P_1 and P_4 of the new curve to P_4 and P_1, respectively, of the remaining curves. Fig. 12 illustrates the new curve. We calculate P_2 and P_3 by

$$P_2 = P_1 + \gamma(P_1 - P_1'), \quad (6)$$

$$P_3 = P_4 + \gamma(P_4 - P_4'), \quad (7)$$

where γ is a constant value. P_1' and P_4' represent the nearest control points from P_1 and P_4 in the remaining curves, respectively. $P_1 - P_1'$ and $P_4 - P_4'$ represent local gradients. Hence, P_2 and P_3 are the points moved along the local gradients from P_1 and P_4, resulting in a smooth curve.

V. Stroke Deployment

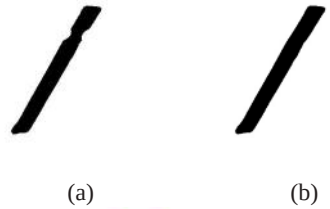
We generate characters by deploying strokes on skeletons. Our character generation method consists of two phases: skeleton modification and stroke deployment.

A. Skeleton modification

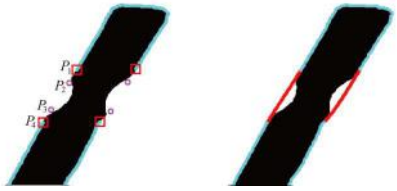
The results of character generation would be strange—even with perfect strokes— if the style of the skeleton dataset were different from the target font. Thus, we modify the skeletons to be similar to the target font. A feasible method for modification is the use of a transformation matrix. We estimate the transformation matrix from the skeletons of the samples. Specifically, we seek two transformation matrices: T_{sz} and T_{aff}. T_{sz} adjusts the size of the skeletons and centroid translation, and T_{aff} adjusts the affine transformation of the skeletons. Modification is carried out by applying T_{sz} and T_{aff} to the dataset.

We estimate T_{sz} from the rectangles of the skeletons. Let H and W denote the height and width of a character in the dataset, respectively. Likewise, let $\hat{H}$ and $\hat{W}$ be the height and width of the skeleton of a sample character, respectively. With output image size I_w and I_h, we calculate T_{sz} by averaging size transformation over the sample characters $\boldsymbol{C} = \{\cdots, c_i, \cdots\}$ as:

$$T_{sz} = \frac{1}{|\boldsymbol{C}|} \sum_{c_i \in \boldsymbol{C}} \begin{pmatrix} \frac{\hat{W}_{c_i}}{W_{c_i}} & 0 & I_w \frac{\hat{W}_{c_i}}{2W_{c_i}} \\ 0 & \frac{\hat{H}_{c_i}}{H_{c_i}} & I_h \frac{\hat{H}_{c_i}}{2H_{c_i}} \\ 0 & 0 & 1 \end{pmatrix}. \quad (8)$$



(a) (b)



(c) (d)

Fig. 11. Restoration of an extracted stroke. (a) Damaged stroke. (b) Restored stroke. (c) Contours removed with cubic Bézier curves; red rectangles represent P_1 and P_4, and pink circles represent P_2 and P_3. Cyan lines represent contours. (d) The created curves, indicated by red lines.

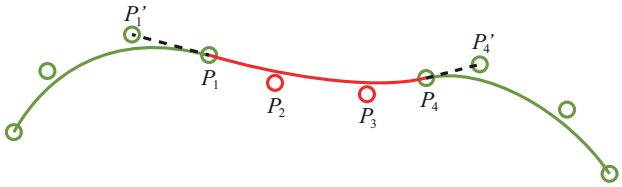


Fig. 12. Points of a new Bézier curve. The red curve represents a new curve and the green curves represent those remaining.

We estimate T_{aff} using the skeletons of the sample and dataset. However, it is difficult to estimate T_{aff} directly from the skeletons because of the complex structure of the characters. If the characters are complex, they have many skeletons. The affine transformations of skeletons cancel each other. Eventually, T_{aff} becomes a trivial matrix, such as an identity matrix. To avoid these problems, we divide the characters into groups and calculate T_{aff} by averaging the affine transformation matrices obtained from each group. We divide a character using the relations of skeletons. Groups consist of skeletons whose relations are continuous, connecting, connected, and crossing. Fig. 13 illustrates the grouping results. We use function f_T to calculate the affine transformation matrix, which fits stroke S to stroke S'. We formulate f_T based on the least-squares method, as Eq. (9), which can be solved analytically as Eq. (10).

$$f_T(S, S') = \arg\min_{T \in \boldsymbol{T}} \|S - TS'\|_2, \quad (9)$$

$$= (S^\top S)^{-1} S^\top S', \quad (10)$$

where $\boldsymbol{T}$ represents a set of all possible affine transformation matrices. We calculate T_{aff} by averaging affine transformation matrices from S to $\hat{S}$ as

$$T_{aff} = \frac{1}{n_{\text{strokes}}} \sum_{i \in \boldsymbol{C}} \sum_{k=1}^{n^i_{\text{groups}}} \sum_{j \in G^i_k} f_T(S^i_j, \hat{S}^i_j), \quad (11)$$

where n_{strokes} is the number of total strokes in $\boldsymbol{C}$. The number of groups in character i is n^i_{groups}. G^i_k is a set of stroke indices in group k of character i, such as: $G^i_k = \{j \mid S^i_j \in \text{group } k\}$.

(a) (b) (c) (d)

Fig. 13. Groups of skeletons: (a) skeleton and (b), (c), and (d) groups.

Finally, we modify the skeletons by applying T_{sz} and T_{aff}. A transformed stroke is obtained using the equation

$$\tilde{S} = T_{aff} T_{sz} S. \tag{12}$$

B. Stroke deployment

Here, we describe the framework for selecting a stroke that is the most suitable for a target skeleton $\tilde{S}$. Stroke deployment has an impact on the appearance of the generated characters; therefore, the appropriate stroke must be chosen for each target skeleton.

First, we select $\hat{S}$, which fits to $\tilde{S}$, from a set of the extracted strokes $\hat{\boldsymbol{S}}$. Then, we determine the stroke $\hat{\mathcal{I}}$ corresponding to the selected $\hat{S}$.

$$\hat{\mathcal{I}} = f_{img}(\hat{S}), \tag{13}$$

$$\hat{S} = \arg\min_{\hat{S}_i \in \hat{\boldsymbol{S}}} E(\hat{S}_i, \tilde{S}), \tag{14}$$

where f_{img} is a function that gives a stroke corresponding to $\hat{S}$ and E is an energy function associated with skeletons $\hat{S}$ and $\tilde{S}$. Once $\hat{\mathcal{I}}$ is determined, we apply $f_T(\hat{S}, \tilde{S})$ to $\hat{\mathcal{I}}$ and deploy the transformed $\hat{\mathcal{I}}$ on $\tilde{S}$. We repeat the selection until strokes for all target skeletons are determined. Finally, a character is generated by integrating the selected strokes.

We define E as

$$E(\hat{S}, \tilde{S}) = E_s(\hat{S}, \tilde{S}) + E_s(f_{data}(\hat{S}), f_{data}(\tilde{S})) + E_a(f_{data}(\hat{S}), f_{data}(\tilde{S})), \tag{15}$$

where $f_{data}(S)$ gives the skeleton corresponding to S in the skeleton dataset. The energy E utilizes three terms in inspecting $\hat{S}$. The first term measures the distance between skeletons in a sample character and a target character. When this term is small, two skeletons are similar. Therefore, $\hat{\mathcal{I}}$ will naturally fit to the target skeleton. The second term also measures the distance between two strokes, but only using the dataset. If the skeletons in the dataset are similar, the stroke is favorable for the target skeleton. This term improves the accuracy of the energy function. The third term measures distance using adjacent skeletons, making the distance more global than when focusing on only two skeletons.

The energy E_s measures the distance between two skeletons. We define $E_s(S, S')$ as

$$E_s(S, S') = \|S' - f_T(S, S')S\|_1 + \|S - f_T(S', S)S'\|_1 + \sum_{k=1}^{3} \mathbb{I}_k(S, S'), \tag{16}$$

where $\mathbb{I}_k$ is 1 if the three attributes of two skeletons—line type, start shape, and end shape—are different; otherwise, it is 0. $\mathbb{I}_k$ inspects the line type, start shape, and end shape of the stroke when $k = 1$, 2, and 3, respectively. The first term of E_s measures the distance between two skeletons. The second term of E_s takes into account the distortion from the transformation $f_T(S, S')$; a less distorted skeleton is favored. The third term of E_s incorporates attribute differences. We define E_a as

$$E_a(S, S') = \begin{cases} \|S_{st} - S'_{st}\|_1 + \|S_{ed} - S'_{ed}\|_1 & \text{if } \boldsymbol{S}_a \text{exists} \\ 50 & \text{otherwise,} \end{cases} \tag{17}$$

where $\boldsymbol{S}_a = \{S_{st}, S_{ed}, S'_{st}, S'_{ed}\}$, and S_{st} represent a skeleton connected to the start point of S. In the case where there are several skeletons connected to S at the start point, we choose one skeleton whose center is closest to S. Likewise, S_{ed} represents a skeleton connected to the end point of S. If S_{st}, S_{ed}, S'_{st}, or S'_{ed} is unfavorable, E_a has a large value.

VI. Sample selection

The proposed method uses $\boldsymbol{C}$, a font subset, to generate a large number of characters. The generation results are deeply influenced by $\boldsymbol{C}$. Therefore, it is important to analyze which characters are suitable elements for $\boldsymbol{C}$. With an optimal $\boldsymbol{C}$, we are able to maximize the capability of the proposed method. We define a process of seeking the optimal $\boldsymbol{C}$ as *sample selection*.

In sample selection, we use the skeleton dataset to seek $\boldsymbol{C}$. Note that sample images are not used. Since the skeletons in the dataset are fundamental data, if the skeletons in $\boldsymbol{C}$ in the dataset are suitable, $\boldsymbol{C}$ in a target font can be expected to work well.

In order to ensure sample selection feasibility, we seek $\boldsymbol{C}$ in a subset of the dataset that we define as *validation characters*. In this study, we adopt the 1,006 characters of kyoiku-kanji containing the elemental characters of Japanese. Since the number of characters in the dataset exceeds 210,000, it is time-consuming to use the entire dataset. As we show in experimental results in Section VII, the proposed method is able to generate acceptable results with a selected $\boldsymbol{C}$, verifying the effectiveness of our approach.

Sample selection is based on a genetic algorithm. There are a large number of candidates of optimal $\boldsymbol{C}$ even with the subset. Candidates undergo crossover, evaluation and selection processes and an optimal candidate can be obtained over many iterations of these processes.

We define function $f_{selection}$ that calculates the energy of a candidate $\boldsymbol{C}_i$ using a set of skeletons $\boldsymbol{S}_i$ of $\boldsymbol{C}_i$. Let $\boldsymbol{S}_v$ represent a set of skeletons of validation characters.

$$f_{selection}(\boldsymbol{S}_i) = \alpha f_e(\boldsymbol{S}_i) + (1 - \alpha) f_r(\boldsymbol{S}_i), \tag{18}$$

$$f_e(\boldsymbol{S}_i) = \sum_{s_v \in \boldsymbol{S}_v} \left(\min_{s \in \boldsymbol{S}_i} E_s(s, s_v) + E_a(s, s_v) \right), \tag{19}$$

$$f_r(\boldsymbol{S}_i) = |\boldsymbol{S}_i| + N_{cross}(\boldsymbol{S}_i) + N_{con}(\boldsymbol{S}_i), |N| \tag{20}$$

where $|\boldsymbol{S}|$ represents the cardinarity. N_{cross} gives the number of skeletons whose relations are crossing. N_{con} gives the number of skeletons whose relations are continuous, connecting, and connected. f_r measures the complexity of $\boldsymbol{C}$ and serves as a regularization term. We control the complexity of the samples with a constant value α.

VII. Experimental results

We demonstrate character generation with the proposed method using kanji. We used five fonts as the target fonts: Ibara 43), Gcomic 44), Onryo 45), Tsunoda 46), and Zinpen 47), where the indices represent the numbers in Appendix. Fig. 14 shows the original characters in the five fonts. The characters in each font have distinctive strokes; for instance, two parallel lines are used in Zinpen. It is very difficult for existing methods to generate characters in these characteristic fonts.

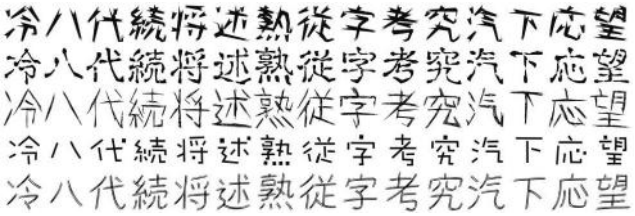


Fig. 14. The Ibara, Gcomic, Onryo, Tsunoda, and Zinpen fonts (from top to bottom).

TABLE II
SAMPLE SELECTION RESULTS.

α	$f_{selection}$	f_r	samples
0.8	0.017	0.673	字近弟院注病芽浅望浴冷勢総従派
0.6	0.027	0.447	下字八汽考究代続望冷応述従熟将
0.4	0.030	0.387	一二八汽鳥究号代続望冷述従将派

We created sample images by drawing characters in a 360-pt font size using the Qt library[9]. The size of the images is 500 by 500 pixels. We implemented the proposed method with C++ and experiments were carried out on a Windows machine with a dual-core CPU. Fifteen characters were chosen as the samples in each font. The coefficients were set as follows: $\beta_1 = 2.0$ and $\beta_2 = 1.5$. We generated 2,965 characters, which comprise the set of JIS level-1.

For comparison, we used the existing method [28] based on patch transformation developed by Saito *et al*. To the best of our knowledge, [28] is the only method that is applicable to characters in various fonts. [28] divides samples into grid squares and generates characters by deploying the squares. The frameworks of the proposed and [28] are similar. Both exploit samples, extract components of characters, and generate characters. However, the extracted components are significantly different. The components in [28] are small pieces of characters that lack meaning, whereas the components in the proposed method are complete strokes.

We have employed our method for sample selection, the results of which are summarized in Table II. We extracted strokes of the validation characters from the 1,006 characters from kyoiku-kanji that are for elementary school students, as determined by the Japanese Ministry of Education. The initial candidates are randomly generated. We fixed the number of elements in a candidate to 15 characters. In each iteration of the algorithm, 20 candidates survive as good candidates and 150 new candidates are created. The maximum number of iterations is set to 1,000. We varied α and carried out sample selection. The minimum $f_{selection}$ and f_r are listed in Table II. The obtained samples are complex characters at high values of α and simpler at low α values. We use samples at $\alpha = 0.6$ as the sample characters in the following experiments.

A. Character generation results

We generated characters in the five fonts using the proposed and existing methods, which are shown in Fig. 15. We generated 2,965 characters in each font, though only 75 characters are shown due to space limitations. Almost all characters generated by the proposed method appear clean and have good readability. Moreover, the font characteristics, such as the slant of the characters in Ibara, are successfully reconstructed. It is not easy to distinguish the original characters from those generated by the proposed method at a glance.

[9]http://www.qt.io/

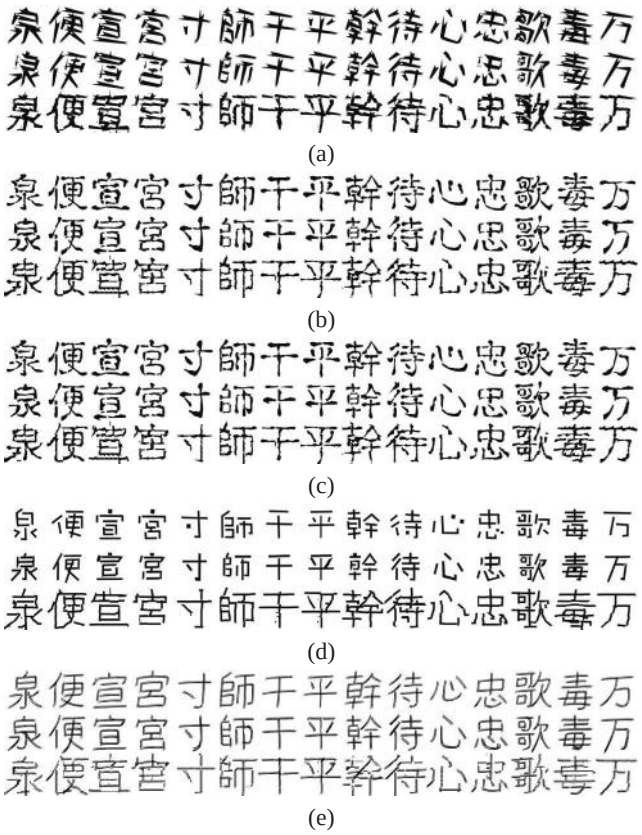


Fig. 15. The characters generated in five fonts: (a) Ibara, (b) Gcomic, (c) Onryo, (d) Tsunoda, and (e) Zinpen. From top to bottom in each subfigure, results are shown for the original characters, the proposed method, and the method of Saito *et al.* [28].

In addition, we describe the complexity of the strokes of the generated characters: 1 (min), 29 (max), 12.5 (avg), and 4.5 (std).

B. Similarity evaluation of the images

We evaluated the generated characters as images. We compared the generated characters with the original images with the Chamfer distance [35]. Let $d_{cham}(A, B)$ be the Chamfer distance from image A to image B. The distance is defined as $d_{cham}(A, B) = \sum_{p \in \hat{A}} \min_{q \in \hat{B}} |p - q|$, where $\hat{A}$ and $\hat{B}$ are the sets of edge points of A and B, respectively. We obtained $\hat{A}$ and $\hat{B}$ by applying a Canny edge detector [36]. In general, $d_{cham}(A, B)$ is not equal to $d_{cham}(B, A)$; therefore, the symmetric formulation is often used

$$d'_{cham}(A, B) = d_{cham}(A, B) + d_{cham}(B, A). \quad (21)$$

We employed d'_{cham} in this study.

A subset of the generated characters is used for this evaluation. Specifically, we used the 1,006 characters mentioned above. We resized the images of the original and generated characters to 100 by 100 pixels and calculated the Chamfer distance between the generated characters and the originals. Table III summarizes the average Chamfer distances. The proposed method is closer for all five fonts than the existing method. In particular, the proposed method is greatly superior to the existing method for the Tsunoda font. Tsunoda has a distinctive skeleton, as can be seen in the spaces between strokes. Since the proposed method successfully modified the skeleton for Tsunoda, the generated characters are close to the

TABLE III
AVERAGE CHAMFER DISTANCE

	Ibara	Gcomic	Onryo	Tsunoda	Zinpen
Saito [28]	4.7	4.2	4.6	6.8	3.8
Proposed	4.1	3.7	4.2	4.1	3.2

TABLE IV
RECOGNITION RESULTS (%)

	Ibara	Gcomic	Onryo	Tsunoda	Zinpen
Saito [28]	29.0	61.5	56.5	7.4	68.8
Proposed	48.3	77.4	67.7	28.6	77.4

originals. These results numerically demonstrate the effectiveness of the proposed method.

C. Evaluation using character recognition

We evaluated the generated characters by using them as training data for a character recognition system. The test data are character images of kyoiku-kanji in each font. Therefore, the number of classes is 1,006. A simple recognition method with Chamfer distance is used in this study. We calculated the Chamfer distance between the training and test data and classified the test data as the nearest character. Test data are created as images, including one JPEG-compressed character. We created test data by drawing characters using a 60-pt font size on 100 by 100 pixel images; the background is white, and the foreground is black. Table IV summarizes the results. The proposed method achieved higher performance than the existing method for all fonts.

D. Subjective evaluation

We present the results of two subjective evaluations based on the similarity of each font to the original characters and the appearance of the generated characters. Both subjective evaluations were carried out by 14 participants.

The first subjective evaluation is of the similarity between the generated and original characters. At the beginning of the first subjective evaluation, we showed the original characters to the participants. Then, 10 idioms consisting of four characters were displayed. The idioms were made from original characters, characters generated by the proposed method, or characters generated by the existing method. We use the mean opinion score (MOS) over the results. The participants assigned scores ranging from 1 (bad) to 5 (excellent) to the idioms according to their impression of their similarity. Table V summarizes the results. The proposed method receives higher MOSs than the existing method. Moreover, the MOSs of the proposed method are relatively close to the originals.

The second subjective evaluation is of appearance. We asked the participants to select characters that may have been generated by computers. The test data consisted of 150 characters from each font, i.e., 50 characters each from the original characters, those generated by the existing method, and those generated by the proposed method. Therefore, the test data consist of a total of 750 characters. Fig. 16 illustrates the test data shown to the participants. We counted the number of participants who believed the characters to have been generated artificially. Then, we calculated the averages and normalized the numbers. Table VI summarizes the results. A character's appearance is natural if the value is low and unnatural if it is high. *Natural* means that the characters are likely handmade, and *unnatural* means artificial. Most of characters generated by the existing method are classified as generated characters since the results are in the range

TABLE V
RESULTS OF A SUBJECTIVE EVALUATION OF FONT SIMILARITY TO ORIGINAL CHARACTERS

	Ibara	Gcomic	Onryo	Tsunoda	Zinpen
Originals	4.5	4.8	4.5	4.9	4.9
Saito [28]	1.1	1.3	1.5	1.1	1.3
Proposed	4.3	4.4	4.6	4.3	4.6

Fig. 16. Examples of test data and answers. (a) Test data shown to the participants. (b) Answers to the test data. Generated characters are gray.

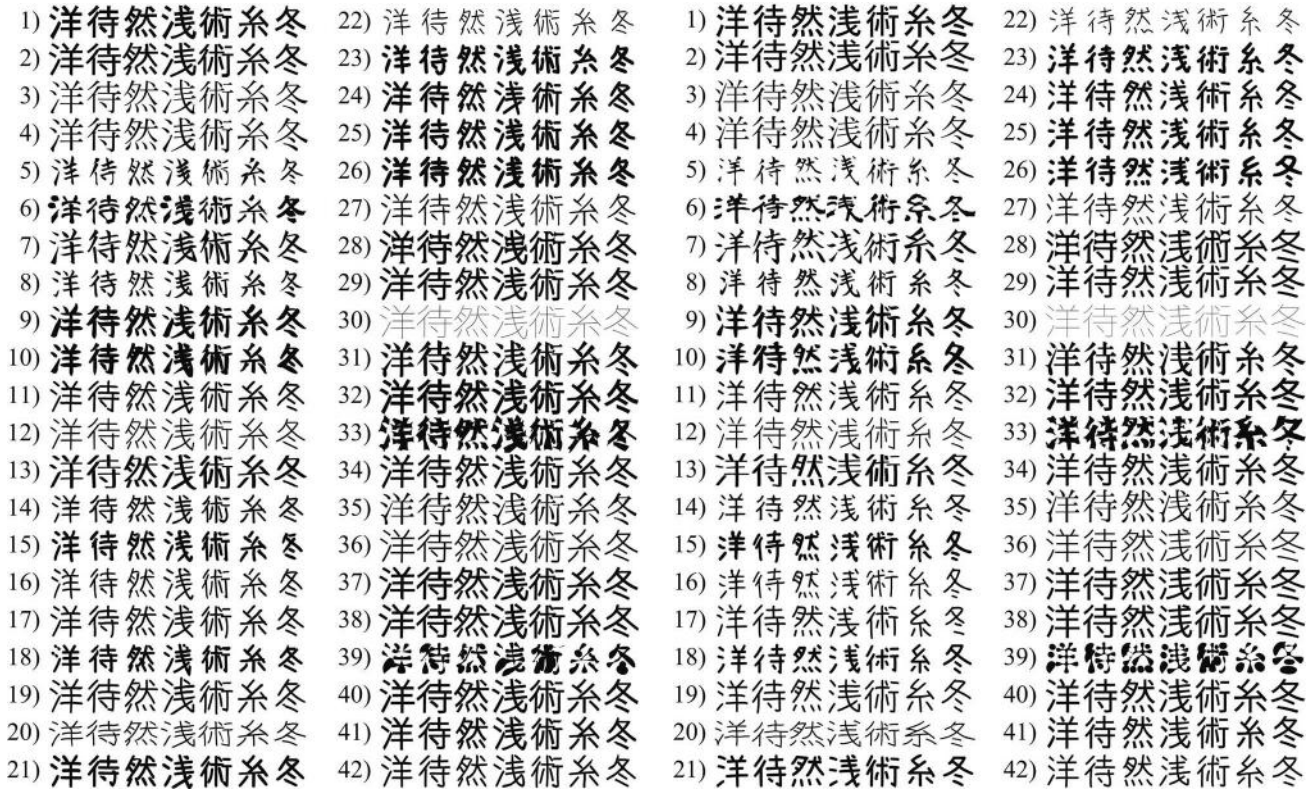
of 0.77–0.99. The appearance of the characters generated by the method in [28] is far from handmade. In contrast, the results for the proposed method are much lower than those for [28]. In particular, the characters generated by the proposed method for Ibara and Onryo are close to the original characters. According to the results, it is difficult to distinguish the original characters and those generated by the proposed method, even by a human. Therefore, the proposed method successfully generated characters with a good appearance.

E. Varied font generation

In order to demonstrate the font generation capability of the proposed method, we performed a generation experiment with 42 fonts: four standard fonts used in Windows (Gothic, Meiryo, Mincho, and YuMincho), six calligraphy handwriting fonts 5) – 10), 17 pen handwriting fonts 11) – 24), and 15 artificial fonts 28) – 42). The fonts are varied, and include Mincho, Gothic, clerical, antique, personal handwriting, professional and handwriting styles. It is worth noting that the number of fonts used in most existing methods is small; in particular, we used a significantly large number of handwritten fonts, 23. In this experiment, we generated 2,965 characters. We illustrate examples of the results in Fig. 17. The results are promising. The proposed method generated clean characters and the style of each font is reproduced.

VIII. CONCLUSION

This paper focused on the problem of generating characters for a typographic font by exploiting a subset of a font and a skeleton dataset. The proposed method extracts character strokes and constructs characters by selecting and deploying the strokes to the skele-



(a) Proposed (b) Original

Fig. 17. Generation results of the proposed method and the original characters in various fonts. Each row shows one font.

TABLE VI
RESULTS OF THE SUBJECTIVE EVALUATION OF APPEARANCE

	Ibara	Gcomic	Onryo	Tsunoda	Zinpen
Originals	0.06	0.02	0.04	0.01	0.01
Saito [28]	0.96	0.9	0.86	0.98	0.94
Proposed	0.11	0.09	0.07	0.13	0.08

ton. The proposed method successfully extracts the natural strokes from sample character images. It is worthwhile to emphasize the importance of stroke extraction from the small number of character images. The proposed method only requires a font subset—as small as 15 samples—which is a feasible number of samples to collect. With such a small subset, the proposed method can generate thousands of characters. Sample collection is further eased because this method can extract characters from image formats.

An experimental evaluation was conducted with five characteristic fonts that are difficult to generate with existing methods. We evaluated the generated characters by objective and subjective evaluations; all results indicated that the characters generated by our proposed method have a comparable appearance, usefulness, and readability to the original characters. Furthermore, we carried out the experiments with subsets of 42 fonts to demonstrate the generative capability of the proposed method. In our future work, we will attempt to automatically adjust skeleton samples and conduct experiments with a larger number and greater variation of fonts.

APPENDIX
LIST OF FONTS

1) MS Gothic
2) MS Meiryo
3) MS Mincho
4) MS YuMincho
5) Aoyagi kouzan
6) Aoyagi reisho
7) Eishi kaisho
8) aiharahude
9) jetblack
10) riitf
11) azukifont
12) chigfont
13) gyate
14) hosofuwafont
15) KajudenB
16) KajudenR
17) kiloji
18) KTEGAKI
19) mitimasu
20) seifuu
21) setofont
22) SNsanafon
23) TAKUMISFONT-B
24) uzurafont
25) apjapanesefonth
26) KFhimaji
27) ruriiro

28) AMEMUCHIGOTHIC
29) antique
30) chogokubosogothic
31) dejima-mincho
32) GenEiAntique
33) hakidame
34) IPAexg
35) IPAexm
36) jk-go-l
37) Kazesawa
38) migu-1c
39) mofuji
40) Nasu
41) Smart
42) yutapon-coding
43) Ibara
44) Gcomic
45) Onryo
46) Tsunoda
47) Zinpen

REFERENCES

[1] S. Bayoudh, H. Mouchère, L. Miclet, and E. Anquetil, "Learning a classifier with very few examples: analogy based and knowledge based generation of new examples for character recognition," in *Machine Learning: ECML 2007*. Springer, 2007, pp. 527–534.

[2] M. Djioua and R. Plamondon, "An interactive system for the automatic generation of huge handwriting databases from a few specimens," in *Pattern Recognition, 19th International Conference on*. IEEE, 2008, pp. 1–4.

[3] Y. Wada, H. Kasuga, and K. Sumita, "An evolutionary approach for the generation of diversiform characters using a handwriting model," in *Pattern Recognition, 16th International Conference on*, vol. 3, 2002, pp. 131–134.

[4] Y. Zheng and D. Doermann, "Handwriting matching and its application to handwriting synthesis," in *Document Analysis and Recognition, Eighth International Conference on*. IEEE, 2005, pp. 861–865.

[5] B. Bai, K.-W. Wong, and Y. Zhang, "An efficient physically-based model for chinese brush," in *Frontiers in Algorithmics*. Springer, 2007, pp. 261–270.

[6] K. W. Kwok, S. M. Wong, K. W. Lo, and Y. Yam, "Genetic algorithm-based brush stroke generation for replication of chinese calligraphic character," in *2006 IEEE International Conference on Evolutionary Computation*, 2006, pp. 1057–1064.

[7] D. E. Knuth, *The METAFONTbook*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[8] E. J. Jakubiak, R. N. Perry, and S. F. Frisken, "An improved representation for stroke-based fonts," in *Proc. of ACM SIGGRAPH*, 2006.

[9] C. Hu and R. D. Hersch, "Parameterizable fonts based on shape components," *Computer Graphics and Applications*, vol. 21, no. 3, pp. 70–85, 2001.

[10] T. Hassan, C. Hu, and R. D. Hersch, "Next generation typeface representations: revisiting parametric fonts," in *Proceedings of the 10th ACM symposium on Document engineering*. ACM, 2010, pp. 181–184.

[11] T. Tanaka, H. Iwasaki, K. Nagahashi, and E. Wada, "Making kanji skeleton fonts through compositing parts," *Transactions of Information Processing Society of Japan*, vol. 36, no. 9, pp. 2122–2131, 1995, in Japanese.

[12] K. Kamichi, "Kage — an automatic glyph generating engine for large character code set," in *Proceedings of the glyph and typesetting workshop*, 2004, pp. 85–92, in Japanese.

[13] S. Xu, F. Lau, W. K. Cheung, and Y. Pan, "Automatic generation of artistic chinese calligraphy," *Intelligent Systems, IEEE*, vol. 20, no. 3, pp. 32–39, 2005.

[14] S. Xu, H. Jiang, T. Jin, F. C. M. Lau, and Y. Pan, "Automatic generation of chinese calligraphic writings with style imitation," *IEEE intelligent systems*, vol. 24, no. 2, pp. 44–53, 2009.

[15] H. Choi, S.-J. Cho, and J. H. Kim, "Generation of handwritten characters with bayesian network based on-line handwriting recognizers," in *Document Analysis and Recognition. Seventh International Conference on*, 2003, pp. 995–999.

[16] R. Suveeranont and T. Igarashi, "Example-based automatic font generation," in *Smart Graphics*. Springer, 2010, pp. 127–138.

[17] N. D. F. Campbell and J. Kautz, "Learning a manifold of fonts," *ACM SIGGRAPH*, vol. 33, no. 4, 2014.

[18] J. chao Feng, C. K. Tse, and Y. Qiu, "Wavelet-transform-based strategy for generating new chinese fonts," in *Circuits and Systems, Proceedings of the International Symposium on*, vol. 4, 2003, pp. IV–337–IV–340 vol.4.

[19] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural Comput.*, vol. 12, no. 6, pp. 1247–1283, 2000.

[20] J. W. Lin, C. Y. Hong, R. I. Chang, Y. C. Wang, S. Y. Lin, and J. M. Ho, "Complete font generation of chinese characters in personal handwriting style," in *IEEE 34th International Performance Computing and Communications Conference*, 2015, pp. 1–5.

[21] A. Zong and Y. Zhu, "Strokebank: Automating personalized chinese handwriting generation," in *Proceedings of the Twenty-Sixth Annual Conference on Innovative Applications of Artificial Intelligence*, 2014, pp. 3024–3029.

[22] Y. Wang, H. Wang, C. Pan, and L. Fang, "Style preserving chinese character synthesis based on hierarchical representation of character," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 1097–1100.

[23] B. Zhou, W. Wang, and Z. Chen, "Easy generation of personal chinese handwritten fonts," in *IEEE International Conference on Multimedia and Expo*, 2011, pp. 1–6.

[24] C. Shi, J. Xiao, W. Jia, and C. Xu, "Automatic generation of chinese character based on human vision and prior knowledge of calligraphy," in *Natural Language Processing and Chinese Computing*. Springer, 2012, pp. 23–33.

[25] P. Liu, S. Xu, and S. Lin, "Automatic generation of personalized chinese handwriting characters," in *Digital Home, Fourth International Conference on*, 2012, pp. 109–116.

[26] X. Qiu, W. Jia, and H. Li, "A font style learning and transferring method based on strokes and structure of chinese characters," in *Computer Science Service System, International Conference on*, 2012, pp. 1836–1839.

[27] S. Ju and J. Shin, "Cursive style korean handwriting synthesis based on shape analysis with minimized input data," in *High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, 2013, pp. 2231–2236.

[28] H. Saito, Y. Sugaya, S. Omachi, S. Uchida, M. Iwamura, and K. Kise, "Generation of character patterns from sample character images," in *IEICE Technical Report. PRMU*, vol. 110, no. 467, 2011, pp. 293–298, in Japanese.

[29] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman, "The patch transform and its applications to image editing," in *CVPR 2008*. IEEE, 2008, pp. 1–8.

[30] "Glyphwiki," http://glyphwiki.org/, in Japanese.

[31] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[33] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[34] P. Selinger, "Potrace: a polygon-based tracing algorithm," http://potrace.sourceforge.net/potrace.pdf, 2003.

[35] G. Borgefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, no. 6, pp. 849–865, 1988.

[36] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.