An IEC-based Support System for Font Design[*]

Tatsuo Unemi and Megumi Soda[†]
Department of Information Systems Science
Soka University
1-236 Tangi-machi, Hachioji, Tokyo 192-8577 Japan
unemi@iss.soka.ac.jp, msoda@intlab.soka.ac.jp

Abstract – *This paper describes our first trial to build a prototype of support system for font design utilizing the technique of Interactive Evolutionary Computation. The target domain is Japanese Katakana constructed from very simple stroke elements. Some parameters for drawing elements are encoded on the genome of each candidate individual. Starting from the initial population of sixteen individuals with random genes, the user can breed his/her favourite fonts through the graphical user interface based on a framework of Simulated Breeding. Each candidate is shown in the sub-window with a sample word arbitrarily set up by the user. We certified that it effectively works to produce a type of expressive glyphs.*

Keywords: Interactive evolutionary computation, design support system, font design.

1 Introduction

The Interactive Evolutionary Computing (IEC) [4] has been recognized as a useful method to support human creativity through many types of artistic and engineering works in this decade. However a lot of fields are not revealed yet though IEC must be useful for them. Font design is one of this kind of fields in which only a few experts can do meaningful jobs so far. The needs of expressive font glyphs are not only for professional graphic designers but also for ordinary people who wants to build his/her own unique web pages. It is valuable to make it easy to design font glyphs as users want.

Some researchers have tried to build simple systems, such as Butterfield and Lewis [2] for Roman Alphabet and Chan Kwai Hung *et al* [3] for Chinese characters. The former takes an approach of deformation from existing font shape for typographical art works, and the latter is still in development process for complicated structure by multiple abstract level representation. Butterfield and Lewis employed a parametric font definition where letters are individually deformed by collections of implicit surface primitives for side effects of 3D computer graphics in "Houdini," a well known commercial-based software for 3D animation. They created some interesting animation movies including deformed letters. These challenges are helpful toward more practical ones, but this field is still in the very early stage though there are many possible approaches for many types of characters and many directions of objectives.

This paper describes our first trial to build a prototype of support system for font design utilizing a technique of IEC. As the first step toward a practical application system, we designed and implemented a possible method for a very simple but complete prototype system. The following sections describe the method we took but doesn't include any performance evaluation of the system because it is very early stage.

2 Font representation

There are many types of characters and glyphs in the world. Some are very complicated and some are simple. We choose Japanese Katakana as the first target because it is relatively simple in both terms of structure and elements.

There are two types of data representation of glyph. The first one is by bitmaps, and the second one is by scalable vectors. The former method is used in small font and is not suitable for large font due to the size of memory needed. In recent years, as the computation power and the resolution of displays and printers improved, the vector representation became to be the popular standard. By this reason, we employ a scalable vector representation for font glyphs here.

To guarantee that the produced glyphs are readable as identical to original letters, we use the fixed strokes as the skeleton for each character and encode the parameters for drawing the elements along the skeleton. The strokes of both skeleton and outline are represented by cubic Bezier curves for convenience of both preparing the skeleton data and programming in Java on MacOS X. We drew a set of strokes for the skeleton data using a drawing tool "Adobe Illustrator" then saved them into files of Scalable Vector Graphics (SVG) [6] format. It

[*]**0-7803-7952-7/03/$17.00 © 2003 IEEE.**
[†]Currently working at TGL, Osaka, Japan.

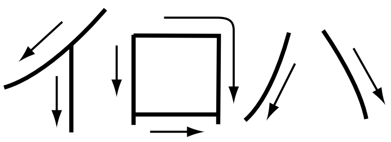

Figure 1: Examples of skeleton strokes of Katakana. These are three letters *i, ro* and *ha* from left to right

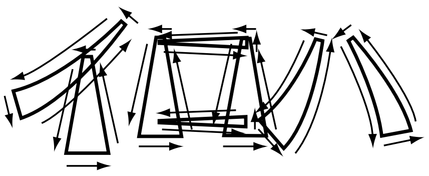

Figure 2: Examples of glyphs as a part of the final product.

is not so difficult to read SVG data to translate them into internal representation on Bezier curves because it follows XML tag format. Figure 1 shows examples of skeleton strokes we used. All of the skeleton data are listed in Appendix A at the tail of this paper. The core tags representing stokes of the left letter "*i*" of Figure 1 in SVG format are:

```
<path d="M32.31,17.322v42"/>
<path d="M48.644,0.322\
   c0,0-25.5,30.833-48.5,37.5"/>
```

The upper tag means a vertical line segment starting from $(32.31, 17.322)$ with the length 42. The lower tag means a cubic Bezier curve starting from $(48.644, 0.322)$ of which co-ordinate of the end point is $(48.5, 37.5)$ relatively from the start point and relative co-ordinate of control points are $(0, 0)$ and $(25.5, 30.833)$. The co-ordinate system of these data is vertically flipped with ordinary mathematical system, that is, the upper left corner is the origin and co-ordinate of vertical axis increase by moving downward.

One segment of Bezier curve is represented by four pairs of 2D co-ordinates, start point, end point and two control points. The trajectory of the curve is defined as following parametoric equation:

$$\vec{b}(t) = (1-t)^3\vec{p}_s + t(1-t)^2\vec{c}_1 + t^2(1-t)\vec{c}_2 + t^3\vec{p}_e \quad (1)$$

for $0 \le t \le 1$ where $\vec{p}_s$ is the start point, $\vec{p}_e$ is the end point, and $\vec{c}_1$ and $\vec{c}_2$ are the control points.

The final product of the system is a font containing a set of glyphs. A glyph, a representation of shape of a letter, is represented a set of outline segments in a form of Bezier curves. The data of this format can easily converted to a scalable font file widely used for personal computers, such as PostScript font. Figure 2 shows examples of glyphs.

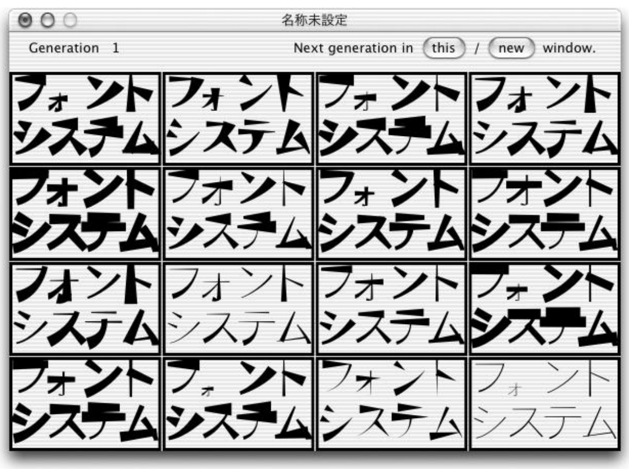

Figure 3: An example of the field window.

3 Basic flow

The system works as follows. Just after the initialization of the system by reading skeletons from disk files written in SVG format into the memory as NSBezierPath [1] instances, the user is prompted to input a sample words in Katakana for visualization. Then the *field* window appears on the screen that contains sixteen sub-windows each of which shows a candidate drawn using an individual genome of the initial population initialized by random genes. The user can breed the glyph on this window in a way of *Simulated Breeding* [5]. Figure 3 shows an example of the field window. As Figure 4 illustrates, the user iterates a cycle of selection and generation change until he/she obtain an acceptable result. The user selects his/her favourite individual(s) from the field window by clicking sub-windows, and pushes the button of "next generation." As the result of these operations, the new population of offspring is generated by mutation and crossover; and the old population is replaced with them if the user chose "in this window" button, or a new field window filled with the offspring is created if "in new window" was chosen. The user can also create a new field window of another initial population in which he/she can operate independent breeding process. Any individual can migrate between different field windows by user's *drag & drop* operation.

The user can examine arbitrary size of sample drawings by using zoom window for any individual in the current field window. The sample words can be arbitrarily changed in any time the user wants. The user may use this system not for creating complete set of glyphs as a new font, but for designing typography of specific words. For this purpose, the user can easily export the produced outline of target words into another graphics tool by *copy & paste* operation from the zoom window.

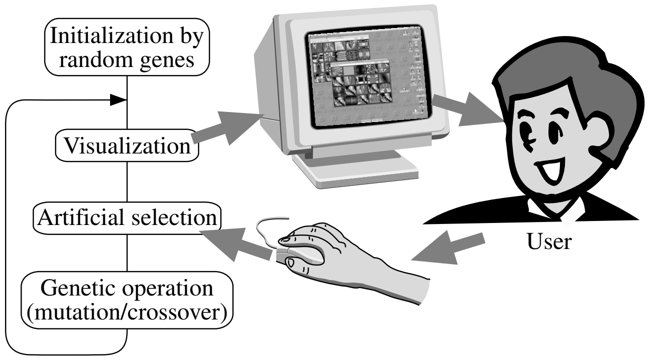
Initialization by random genes
Visualization
Artificial selection
Genetic operation (mutation/crossover)
User

Figure 4: Flow of simulated breeding.

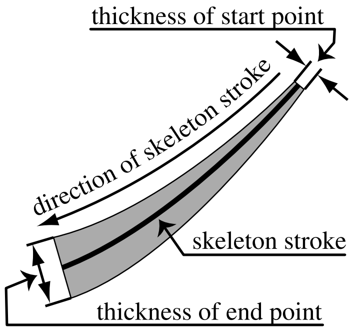

Figure 5: Example of glyph element produced from a skeleton stroke.

4 Morphology

The one of key issues to develop a successful IEC application for a new field is the method to parametarize the target object. In other words, we must define (1) gene coding, (2) morphology, and (3) visualization. The gene code should express rich space of target beyond the users' imagination, and simultaneously should restrict the space within the range of acceptable candidates. It is difficult to cover whole of the possible space, so it is helpful to examine a various types of alternative approaches for improvement of this field.

We designed a prototype genome consisting of a number of floating point numerical values concerning three kinds of phenotypic features as follows.

4.1 Thickness of elements

The first three genes are for the thickness of each element stroke. They are mapped onto the parameters of

(i) thickness of the start point,

(ii) thickness of the end point, and

(iii) thickness ratio by angle;

for the phenotype. Figure 5 shows the effect of the genes (i) and (ii). Each skeleton stroke is transformed into a shape represented by four connected segments as the outline where the thickness at start and end points are calculated from genetic information. Each value of all of genes are with in the range $[0, 1]$, and the concrete value of thickness is calculated by multiplying the gene value, the predefined maximum value of thickness, and the ratio by angle. The ratio by angle is to vary the thickness depending on the absolute angle of line connecting between the start point and the end point, and the value R_a is calculated by

$$R_a = 1 - (1 - \gamma^{G_a}) \cdot \angle(\vec{p}_e - \vec{p}_s) \qquad (2)$$

$$G_a = \begin{cases} 2g_3 - 1 & \text{if } 0.5 < g_3 \leq 1 \\ 1 - 2g_3 & \text{if } 0 \leq g_3 \leq 0.5 \end{cases} \qquad (3)$$

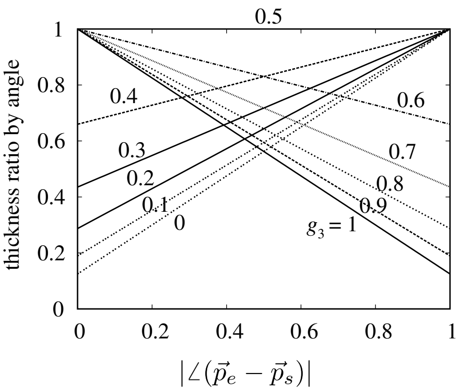

Figure 6: Relation between angle and thickness ratio where $\gamma = 1/8$.

where γ is a constant of $(0, 1)$ and g_3 is the gene value of (iii) listed above. Figure 9 illustrates the relation between angle and thickness ratio where $\gamma = 1/8$. The ratio is always proportional to the angle. The ratios for vertical and horizontal elements are 1 and γ respectively when $g_3 = 0$. The ratio for any angle is 1 when $g_3 = 1/2$. The ratios for vertical and horizontal elements are γ and 1 respectively when $g_3 = 1$.

4.2 Shrinking ratio of small characters

The forth parameter is the shrinking ratio of small letter against normal letter. Usually twelve small letters are used in Katakana set specified in the character code system of Japanese Industrial Standard. It is sometimes useless but has important effect when the sample string includes small letters. The value of shrinking ratio is

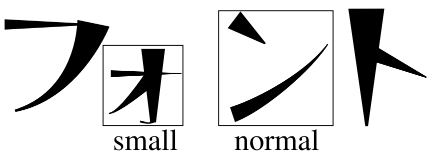

Figure 7: Example of normal and small letters.

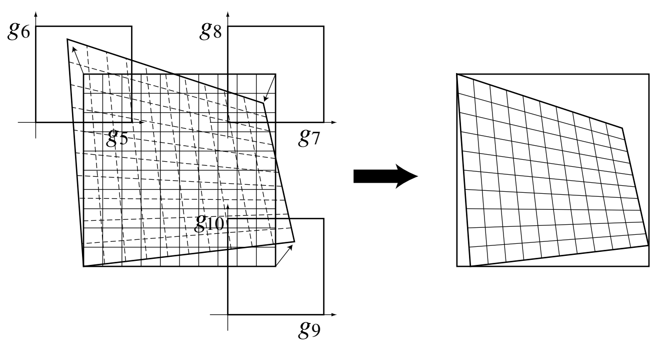

Figure 8: Example of linear deformation of co-ordinates.

calculated by linearly transforming from the range $[0, 1]$ of gene into $[0.3, 0.9]$ for coefficient because it is difficult to distinguish small letters from normal ones if the ratio is greater than 0.9 and the letter is too small to read if it is less than 0.3.

4.3 Deformation of skeleton strokes

The other type of parameters we examined are for deformation of skeleton strokes. There are many possible filter of two dimensional vector graphics, but we implemented only the linear shifting of co-ordinates with six real number parameters. Figure 8 illustrates how these parameters deform the original strokes. First two parameters of six correspond to the vector of shifting the upper left corner of original co-ordinate system. The second two and final two respectively concern the upper right and the lower right corners. If the size of original frame is assumed to be 1 by 1, the transformed co-ordinate (u, v) from (x, y) is calculated by

$$\begin{aligned} u &= \frac{1}{\sigma}(x_1(y - xy) + x_2xy + x_3(x + xy) - o_x) && (4)\\ v &= \frac{1}{\sigma}(y_1(y - xy) + y_2xy + y_3(x + xy) - o_y) && (5)\\ \sigma &= \max(\max(x_2, x_3) - o_x, \max(y_1, y_2) - o_y), && (6) \end{aligned}$$

where $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) are respectively the co-ordinates of upper left, upper right and lower right corners of the transformed frame, $o_x = \min(0, x_1)$, and $o_y = \min(0, y_3)$. The co-ordinate of each corner is calculated by adding modified gene value to the original position as follows.

$$\begin{aligned} (x_1, y_1) &= \alpha(2g_5 - 1, 2g_6 - 1)\\ &\quad +(-\alpha/2, 1 - \alpha/2) && (7)\\ (x_2, y_2) &= \alpha(2g_7 - 1, 2g_8 - 1)\\ &\quad +(1 - \alpha/2, 1 - \alpha/2) && (8)\\ (x_3, y_3) &= \alpha(2g_9 - 1, 2g_{10} - 1)\\ &\quad +(1 - \alpha/2, -\alpha/2). && (9) \end{aligned}$$

where α is a constant of $(0, 1]$ for the width of shifting. The value 0 makes no effect for any gene value. If $\alpha > 1$, it possibly produces twisted letters difficult to read in some combination of gene values. We employed $\alpha = 1$ in our current implementation determined through several times of trials with a variety of values of α.

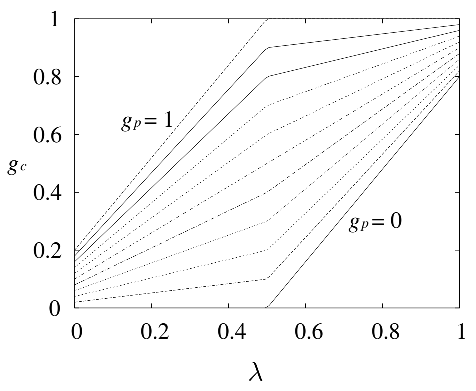

Figure 9: Relation between random number λ and the result value g_c where $\mu = 0.8$.

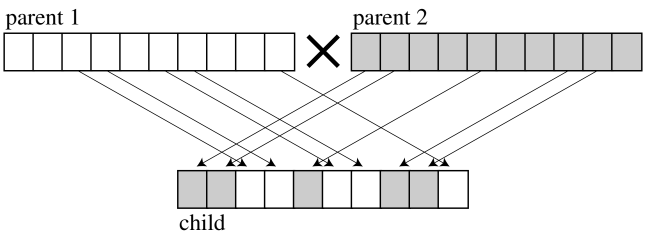

Figure 10: Crossover operation by combining genes from randomly selected parent.

5 Mutation and crossover

As described before, genome of the selected individual mutates to produce its offspring when only one individual was selected. If there are more than one individuals selected, then crossover operation is applied instead of mutation. In both cases, the data structure of genome as the object of the operations is a vector of floating point numbers where each element value is within $[0, 1]$. To make the search by genetic operations, selection, mutation and crossover, to be effective, we implemented the mutation by adding a random portion toward the value boundary of ether lower or upper, as represented by the equation

$$g_c = \begin{cases} g_p - \mu(1 - 2\lambda)\, g_p & \text{if } \lambda < 0.5 \\ g_p + \mu(2\lambda - 1)(1 - g_p) & \text{otherwise} \end{cases} \qquad (10)$$

where g_p is the gene value of the parent, g_c is the gene value of the child, λ is a random real number within $[0, 1]$, and μ is a constant within $[0, 2]$ that specifies the width of mutation. Figure **??** shows the relation between the random number λ and the result value of mutant g_c for various values of g_p when $\mu = 0.8$.

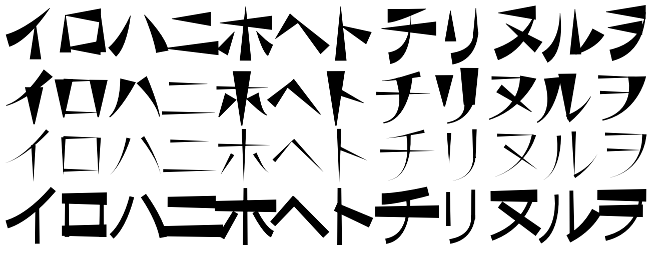

Figure 11: Sample fonts bred with the IEC system without skeleton deformation.

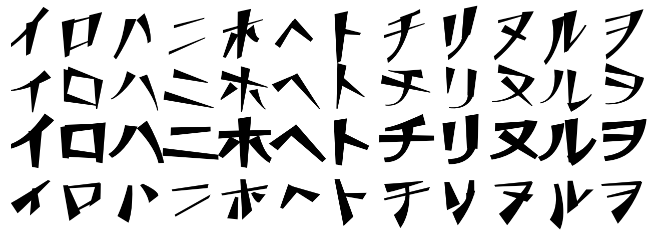

Figure 12: Sample fonts bred with the IEC system with skeleton deformation.

Crossover is done by combination of the parts of genomes from parents. One of two parents is randomly chosen for each gene as the source. Any gene, a floating point number of single precision, never divided into bits; otherwise it might cause an error of numerical calculation. Figure 10 illustrates this operation.

6 Results

Through several times of experimental use by the authors and other students, we certified that the prototype system works well and is able to produce many types of fonts. But the most noticeable drawback we found is the poorness of the search scape. IEC technique is useful for a multi-dimensional space and/or open-ended space where it is difficult for a human to explore efficiently, but this prototype has only ten parameters. In spite of this drawback, we could obtained several types of fonts of a various kinds of impression even if the effects on deformation of skeleton strokes are omitted as shown in Figure 11. Figure 12 shows the effects of deformation.

7 Future extension

There are a lot of candidates for parameters we should introduce as the next improvement.

1. *Non-linear deformation*, such as wave form, circular shape, partial expansion and shrink, and so on, similar to Butterfield and Lewis' approach may be an easy and effective direction of extension.

2. *Shape of the start and end edges of strokes* can be alternated. The easiest extension might be introduction of round shape.

3. *Ornaments of the start and end edges of strokes* are also useful to extend the richness of product candidates. One of the typical ornaments is *serif* used in Times Roman font.

8 Conclusion

We proposed an alternative approach for IEC application for font design. By dividing the representation of a glyph into skeleton and elements, it becomes easy to keep the result phenotype as a character and simultaneously to enrich the search space.

The work for enrichment remains as our future task, but it will be a promising starting point for more flexible system by adding the other features for ornamenting the elements and deforming the skeletons.

We built a simple prototype of font design system utilizing IEC technique. The method to represent a glyph in the combination of structure and elements is useful for IEC application.

As our future works, we will extend and examine the prototype by adding more features on parameters and by applying another type of characters such as Roman Alphabet, Japanese Hiragana, Korean Hangul, and others.

References

[1] Apple Computer Inc., 2003, Application Kit Java API Reference: NSBezierPath, *Cocoa Developer Documentation.*

[2] Butterfield, I. and Lewis, M., 2000, Evolving Fonts, `http://www.accad.ohio-state.edu/~mlewis/AED/Fonts/`

[3] Chan Kwai Hung *et al*, 2002, Chinese Font Designing with Evolutionary Techniques, `http://people.sd.polyu.edu.hk/~sdkhchan/my_project_1.html`

[4] Takagi, H. 2001, Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation, *Proceedings of the IEEE*, Vol. 89, No. 9, pp. 1275–1296.

[5] Unemi, T., 2003, Simulated Breeding – a Framework of Breeding Artifacts on the Computer, *Kybernetes*, Vol. 32, No. 1/2, pp. 203–220.

[6] W3C, 2001, Scalable Vector Graphics (SVG) 1.0 Specification, *W3C Recommendation*, REC-SVG-20010904.

A Skeleton strokes of Katakana