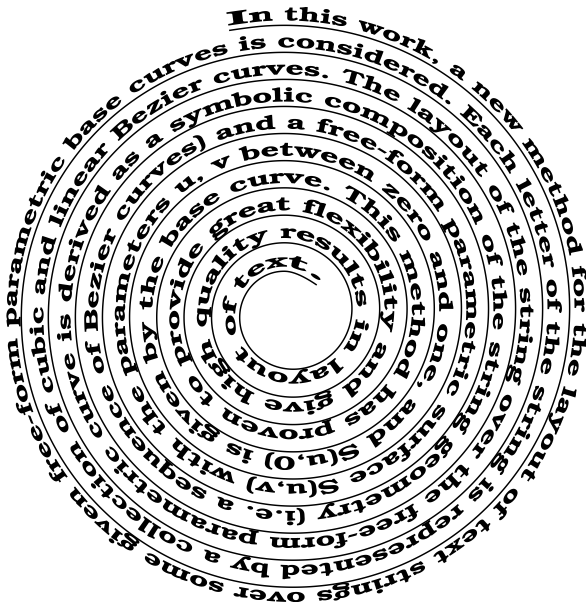# Arbitrary Precise Orientation Specification for Layout of Text

Tatiana Surazhsky
Applied Mathematics Department,
The Technion — Israel Institute of Technology,
Haifa 32000, Israel.
tess@cs.technion.ac.il

Gershon Elber
Faculty of Computer Science,
The Technion — IIT,
Haifa 32000, Israel.
gershon@cs.technion.ac.il

Abstract

In this work, a new method for the layout of text strings over some given free-form parametric base curves is considered. Each letter of the string is represented by a collection of cubic and linear Bézier curves. The layout of the string over the free-form parametric curve is derived as a symbolic composition of the string geometry (i.e. a sequence of Bézier curves) and a free-form parametric surface $S(u,v)$ with the parameters u, v between zero and one, and $S(u,0)$ is given by the base curve. This method has proven to provide great flexibility and give high quality results in layout of text.

1 Introduction

The availability of a whole variety of electronic printing tools and devices have changed the role of typography. Nowadays, text manipulation functions are an integral part of a large body of applications such as multi-media publishing, computer animation and computer-aided design systems.

Computer aided font design tools have been created in 1960s (see [18]), but even today font design systems necessitate the tedious support of a human artist. There is a whole body of work that introduced improvements and extensions into font design systems [5–7, 14, 17, 21] that include programmable parametric models [15]. Some work investigated the extraction of strokes from the outline description of the fonts (see [8, 17]).

A *font* is typically defined as a set of printable or displayable *text characters* of specific style and size. A design of a set of fonts is called *typeface*. Popular type-faces in use today are the *True Type* used by Microsoft Windows [2] and the Adobe's *Type 1* fonts [3]. They are representatives of *scalable* or *vector fonts* which are also called *outline fonts*. Each character of such fonts is described by vector geometry and can be scaled with ease. The curves between the end-points of the vectors are usually specified by using either cubic or linear *Bézier spline curves*. While the most obvious relevance of text characters is for printing and publications, other applications such as computer animation and computer aided design require text manipulation functions.

In [16], one may find a two dimensional morphing method in which each elementary sub-screen shape is defined by a letter shape toward half-toning. This technique is applied in artistic screening which incorporates both full size and microscopic letters into the image reproduction process. In order to avoid counterfeiting, banknotes may include half-toning images with intensity levels that are produced by micro-letters of varying size and shape. Toward this end, the US treasury protects banknotes by using micro-printing techniques for generating letters along curved contours. Many other computerized systems employ printing methods that layout text strings along smooth lines and curves. Such algorithms could be found in systems which employ outline fonts: for example the PostScript language [1,4] and Microsoft Office (the "WordArt" package).

Existing work that allows text deformations, performs the deformations using one of two methods. The first approach defines a best suited rigid motion transformation for each symbol (see pages 171 – 173, "Program 11/Placing Text Along an Arbitrary Path" in [1], for example). This approach is also one of the methods used by the "WordArt" package. The second common technique maps the control points of the Bézier curves comprising the text symbols as in [16]. This second method is better than the first one, since it is more accurate and if the curve is approximated well enough by its control polygon, i.e. when there are enough control points, the deformed letters will look better. Nevertheless, both methods do not guarantee that the letters after
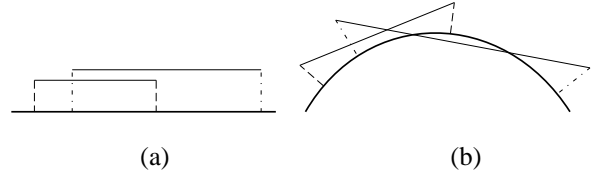
Figure 1. Undeformed linear edges of some character with a base line (thick line) are shown in (a) and do not intersect each other. The deformed base line with the mapped end control points and edges are shown in (b). The straight line edges intersect each other in (b) and the shape even expands below the base line. Note that the lengths of the dashed lines (that are normal to the base line) are unmodified.
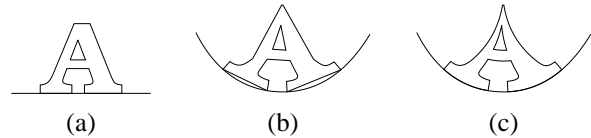
Figure 2. Letter 'A': the original character is shown in (a), with the mapping of the control polygons in (b). The deformation of the symbol using the presented algorithm is shown in (c). Base lines are shown in thick lines.

the deformation will be intersection free. For the second method such intersection artifacts are not frequent, but still exist, see Figure 1 for example. There is another problem with this second, control point mapping, approach. All the linear segments of the symbol stay linear, with possible displeasing artifacts that affect the continuity of adjacent segments. A simple example of this disturbing behavior may be seen in Figure 2. In (a), we find the original geometry of the letter 'A' that has several linear segments. In (b), only the control points of the Bézier curves comprising the symbol 'A' are mapped, as in [16]. Finally, (c) deals with the complete geometry of the curves as is presented in this work. None of the existing tools that the authors are aware of, can produce character deformation with the quality and precision that is presented in Figure 2 (c).

In [20], a free-form deformation technique is presented for solid geometric models. Given a mapping $\mathcal{M} : I\!R^3 \longrightarrow$

2

$I\!R^3$ and object $\mathcal{O} \subset I\!R^3$, $\mathcal{O}$ can be warped to follow $\mathcal{M}$ as $\mathcal{M}(\mathcal{O})$. $\mathcal{M}$ provides a precise control over the warping process. The technique of [20] resembles our method presented herein, yet in a higher dimension; A volume is deformed in [20], while a planar surface is deformed here. A deformation is specifically applied to font and text design and manipulation, in this work. This paper is organized as follows. In Section 2, the basic algorithm is discussed. In Section 3, some possible extensions and applications of the described method are discussed. Examples may be found in Section 4. In Section 5, the problem of feeding the result back into PostScript, creating a closure, is considered. Finally, we conclude in Section 6.

2 Layout of text

Recall that outline font symbols are usually represented by linear or cubic Bézier curves. Bézier curves are variation diminishing and affine invariant [12], which makes them suitable for geometric design purposes and ideal choice for scalable font representation. One more attractive aspect of Bézier spline curves is that they are defined by finite number of control points. Two points identify a linear spline and four points are sufficient for the definition of a cubic Bézier.

Our goal is to layout text along a certain free-form parametric curve $\gamma(t)$, denoted the *base curve*. An exact solution that defines the space for the placement of the text and precisely relocates the geometry of the symbols is presented in Sections 2.1 and 2.2.

2.1 Deformation of text

A natural precise solution for text deformation is to use symbolic composition [9, 10], that is to be calculated for every spline curve $C(t) = \{c_x(t), c_y(t)\}$. Evaluate,

$$C(t) \longrightarrow S \circ C(t) = S\left(c_x(t), c_y(t)\right), \qquad (2.1)$$

while the original symbols lie in the parametric domain of the surface $S(u, v)$.

We exploited the functionality of the IRIT [13] solid modeler to carry out all the necessary symbolic composition computations between Bézier curves and surfaces, fol-

lowing [9]. Nevertheless, the base curve is defined as a B-spline curve. One may convert a B-spline surface that is derived from a B-spline base curve into a set of Bézier surfaces by subdividing the B-spline surface at all its internal knots. Further, by subdividing all the Bézier curves that prescribe the geometry of the character at the corresponding knot values, one is able to complete the deformation process as a composition between the groups of Bézier curves with the corresponding subdivided Bézier patches.

All linear and cubic Bézier curves of the letters are in the parametric space of $S(u, v)$ and are to be subdivided at the certain interior knot value u_0 (or v_0) of $S(u, v)$. Subdividing a curve at the coordinate $u_0 \in [0, 1]$ equals to solving $c_u(t) = u_0$ for all t that satisfy the equation. Herein, the solution set may be found analytically, solving either a linear or a cubic equation and considering only real roots in the range $[0, 1]$.

2.2 The placement of the text

Let $S(u, v)$ be a free-form parametric surface $S(u, v)$, such that given a base line curve $\gamma(t)$,

$$\gamma : [0, 1] \longrightarrow I\!R^2; \quad \gamma(t) = \{x(t), y(t)\}, \qquad (2.2)$$

then

$$S : [0, 1] \times [0, 1] \longrightarrow I\!R^2; \quad S(u, v) = \{x(u, v), y(u, v)\};$$
$$(2.3)$$

$$S(u, 0) = \gamma(u). \qquad (2.4)$$

Note that $\gamma(u)$ and $S(u, v)$ need not be planar. Nevertheless, here we discuss curves located in the plane since we are dealing with two dimensional printed pages.

Consider the bounding box that contains the text string shown on Figure 3 (a) and let $\gamma(t)$ be the base line to place the text along (see Figure 3 (b)). We would like to map the two dimensional rectangular area shown in (a) to a surface defined by some function $S(u, v)$ so that the base line unites with $\gamma(u) = S(u, 0)$. One result can be seen in Figure 4.

The composition function that provides flexibility on one side and precision on the other has been discussed in Section 2.1. Let us return to the definition of the mapping surface $S(u, v)$. The base line $\gamma(t)$ prescribes $S(u, 0)$. Con-
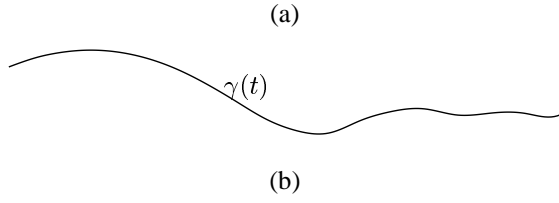
Figure 3. Linear strip containing the given text (a) and the base line $\gamma(u) = S(u,0)$ **of surfaces** $S(u,v)$ **in (b).**

Figure 4. Layout of the string along the curve.

Figure 5. The self-intersecting planar ruled surface between a base curve $\gamma(t)$ **and its translated upward version,** $\widetilde{\gamma}(t)$**.**

sider the upper boundary curve of $S(u,v)$, i.e. $S(u,1)$. With both curves characterized, one can define $S(u,v)$ as a ruled surface between them:
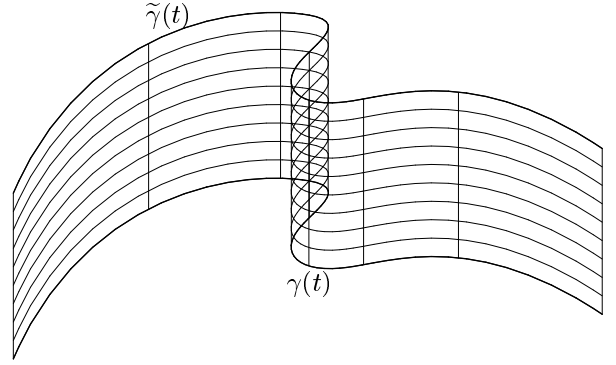
$$S(u,v) = S(u,0)(1-v) + S(u,1)v. \qquad (2.5)$$

One appealing option could be to choose $S(u,1)$ as an offset [11] of the given base curve $\gamma(u)$. Another option could be a vertical translation of $\gamma(t)$.

Let $\gamma(t)$ be a C^1 continuous curve and assume $\gamma(t)$ is either monotone with respect to some line or is closed. Then,

Proposition 2.1 *Let* $\tilde{\gamma}(t)$ *be the offset curve of* $\gamma(t)$*. Suppose that the curves* $\tilde{\gamma}(t)$ *and* $\gamma(t)$ *are simple and mutually intersection free. Then, the ruled surface* $S(u,v) = \gamma(u)(1-v) + \tilde{\gamma}(u)v$ *is self intersections free as well.*

The proof of this proposition may be found in [19], since the tangent vectors of both curves $\gamma(u)$ and $\tilde{\gamma}(u)$ are collinear throughout the parametric domain of u. Thus, if the offset curve of $\tilde{\gamma}(u)$ is simple and do not intersect $\gamma(u)$, the resulting text would have no fold-overs.

While rare, the constraint for $\gamma(t)$ to be either closed or monotone is sufficient but unnecessary. This constraint prevents the singular case for which the end points of $\gamma(t)$ and the end points of $\tilde{\gamma}(t)$ are interleaved, resulting in two simple curves that do not intersect whereas the ruled surface between the curves does self intersect.

The prevention of self intersections in the isoparametric curve $S(u,1)$ could also be materialized by selecting $S(u,1)$ to be a translated version of the base curve $\gamma(t)$. Of course, this solution could yield self intersections in $S(u,v)$ (see Figure 5 for example), and hence, to possible intersections in the composed text.

3   Possible Extensions

One application of the presented method has been described earlier in Section 1, that is, to layout a text string along a given parametric curve. Consider the surface $S(u,v) = \gamma(u) + \delta(v)$ where $\delta(v) = S(0,v)$ is a vertically oriented Bézier curve denoted the *shape line*, while $\gamma(u) = S(u,0)$ is the *base line* curve as before. In this case, the base line of the given text follows $\gamma(u)$, but the font's shape is following $\delta(v)$ (see examples in Figure 6).

The selected base line for the text should be parameterized by its *arc-length*. Otherwise, the end result could be a displeasing mapping of text with varying width (compare Figure 4 with Figure 7). Since the arc-length is not a rational function, we employ an approximation of the arc-length, via a reparameterization of the base line curve, following [10].

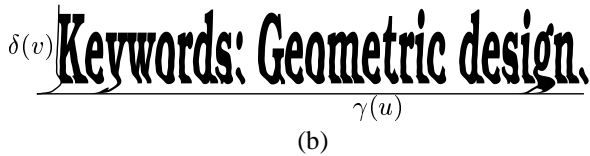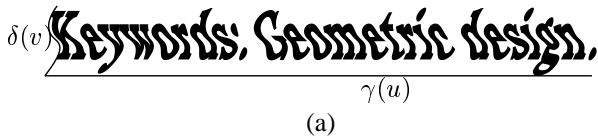In contrast, one could intentionally employ a non arc-

4

$\delta(v)$

$\gamma(u)$

(a)

$\delta(v)$

$\gamma(u)$

(b)

Figure 6. Font altering using a layout over different vertical shaping curves.

Figure 7. Layout of the string along the base curve without arc-length reparameterization. Compare with Figure 4.

length parameterization of the base line curve in order to emphasize some substring in the given text. See Figure 8 for an example.

4   More Examples

Consider the following simple examples, illustrating the applications of the presented technique. Figure 4 illustrated the application of the presented method to the case described in Section 2.2 of an offset curve (Using the input of Figure 3). The following three text strings, that are

Figure 8. An application of an intentional non arc-length parameterization of the base line curve, toward the possible emphasis of certain words. Compare with Figure 4.
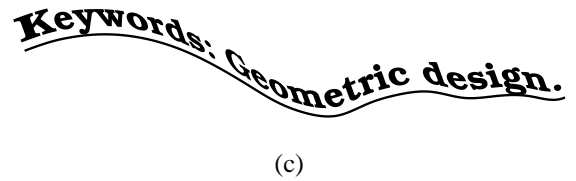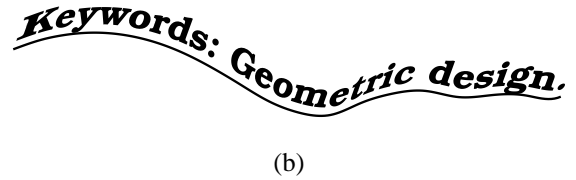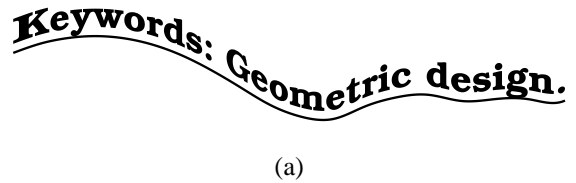

(a)

(b)

(c)

Figure 9. Layout of the string along free-form base curve using a vertical (a) and non vertical ((b) and (c)) translation.

shown in Figure 9, represent the text mapping to the surface $S(u, v)$ where the isoparametric curve $\gamma(u) = S(u, 0)$ is the same base line, and $S(u, 1)$ is its translated version. In (a), the translation is in the vertical direction, in (b) the translation is slightly shifted to the right, and in (c) to the left, creating slanting effects.

The example in Figure 7 shows a simple version of the layout presented in Figure 4 only without the arc-length reparameterization of the base curve $\gamma(u) = S(u, 0)$. The pictures in Figure 6 and Figure 10 show the possibilities of font shaping with the aid of different $\delta(v)$ function.

In Figure 11, one can examine the result of a highly curved base line applied to the symbols 'A' and 'C' in (a) and to the characters 'A' and 'U' in (b). Compare the shape of the deformed letters with the original symbols in Figure 12 (a) and (b), respectively.

All the examples presented in this work were computed in less than a second on a modern Pentium based machine, except the helical abstract example in the first page that took about a dozen seconds to compute on the same machine.

5

Figure 10. New font variation using a specific shape line.

(a)   (b)
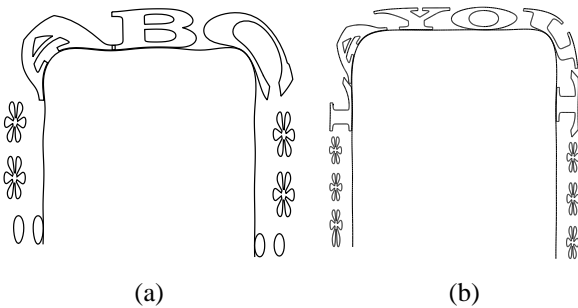
Figure 11. String layout over highly curved lines (a) and (b). Compare with the original strings in Figure 12.

(a)

(b)
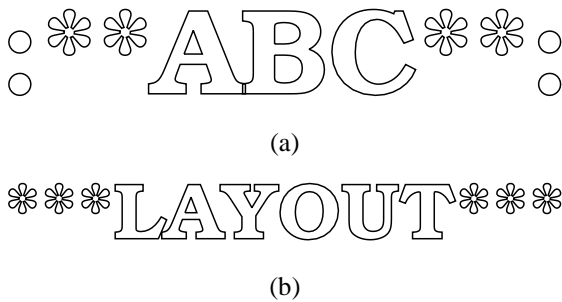
Figure 12. The original strings that appear deformed in Figure 11.

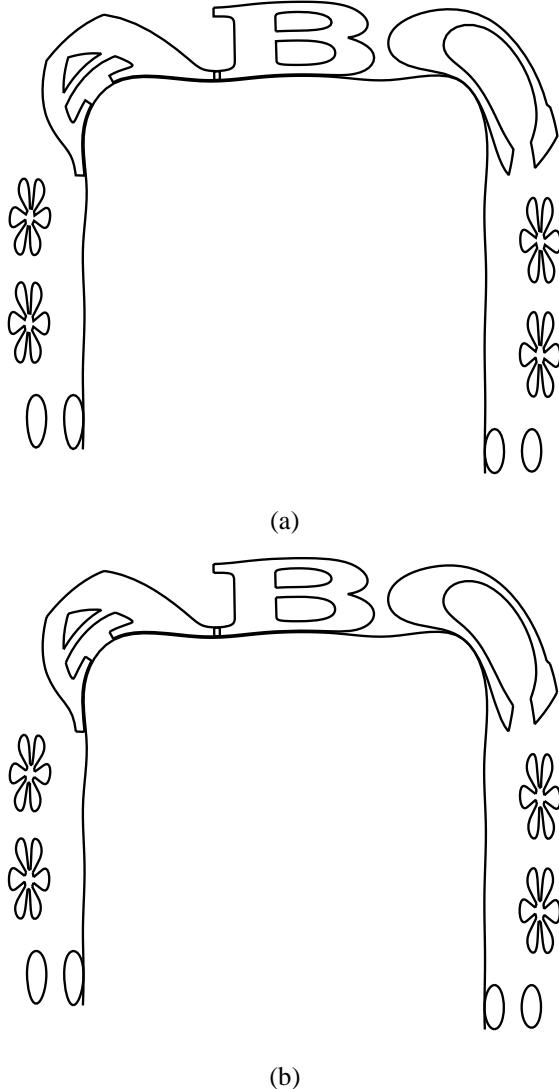## 5 PostScript representation

In our work, we have used *precise* composition between the linear or cubic Bézier curves and arbitrary Bézier/B-spline surfaces. Thus, the order of the resulting curves might be higher than 3 (cubic). Recall that PostScript language supports either linear, or cubic Bézier curves. One could approximate the geometry of the composed letters by a set of cubic Bézier curves to an arbitrary precision [9], in order to further use the composed letters in the PostScript representation.

For example, in Figure 13, two representations of the same string over the same curve are shown. The surface $S(u,v)$ is cubic by linear resulting in composed curve of degree 12 for cubic Bézier and of degree 4 for linear Bézier curve segments. The curves in (a) are approximated by piecewise linear segments whereas the same curves in (b) are represented by cubic Bézier curves. The number of linear segments for each high order curve is equal to seven in this example. The maximal number of cubic Bézier segments approximating a higher order curve that was received as a result of the composition of the outline font representation with the surface patch is two.

6 Conclusions

In this work, we have introduced a precise technique for layout of text along free-form parametric curves. In order to achieve a nicely looking text deformation, the geometric representation of the text (linear and cubic Bézier curves) is symbolically composed with the parametric surface constructed along the given base curve. Moreover, this method may be applied for the purpose of font shaping as well, as was demonstrated in Sections 3 and 4.

We expect to use the described method for some related applications. For instance, text may now be easily animated. Since text could be mapped to *any* simple surface, it could be mapped to each surface in a surface-morphing sequence between, for example, two given key shapes. Alternatively, the base line may be metamorphed and the text may be placed along the curves of the metamorphosis sequence.

(a)

(b)

Figure 13. Two different representations of the same string over the same curve. (a) is approximated by piecewise linear segments, (b) is represented by cubic Bézier curves.

References

[1] *PostScript Language Tutorial and Cookbook.* Adobe Systems Incorporated. Addison-Wesley Publishing Company, 1989.

[2] *The True Type font format specification.* Microsoft Corporation, 1990.

[3] *Adobe type 1 font format: multiple master extensions.* Adobe Developer Support, 1992.

[4] *PostScript language reference manual.* Adobe Systems Incorporated. Addison-Wesley Publisher Company, second edition, November 1994.

[5] D. Adams and J. Andr´e. New trends in digital typography. *Raster Imaging and Digital Typography*, pages 14 – 21, 1989.

[6] H. Changyuan and Z. Fuyan. Automatic hinting of chinese outline font based on stroke separating method. *Proceedings of the First Pacific Conference on Computer Graphics and Applications – Pacific Graphics '93*, 1:359 – 368, 1993.

[7] P. Coueignoux. Generation of roman printed fonts, Ph.D. thesis. 1975.

[8] M. J. Dürst. Structured character description for font design: a preliminary approach based on prolog. *Pacific Graphics '93*, 1:369 – 380, 1993.

[9] G. Elber. Free form surface analysis using a hybrid of symbolic and numeric computation. Ph.D. dissertation. 1992.

[10] G. Elber. Symbolic and numeric computation in curve interrogation. *Computer Graphics forum*, 14(1):25 – 34, March 1995.

[11] G. Elber and E. Cohen. Error bounded variable distance offset operator for free form curves and surfaces. *Int. J. Coput. Geom. Appl. 1*, 1:67 – 78, March 1991.

[12] G. Farin. *Curves and surfaces for computer aided geometric design.* Academic Press, Inc., third edition, 1993.

[13] IRIT. *Irit 7.0 User's Manual.* Technion, Israel. http://www.cs.technion.ac.il/˜irit., Mar. 1997.

[14] P. Karow. Automatic hinting for intelligent font scaling. *Raster Imaging and Digital Typography*, pages 232 – 241, 1989.

[15] D. E. Knuth. *The METAFONT book.* Adison Wesley, 1986.

[16] V. Ostromoukhov and R. D. Hersch. Artistic screening. *SIGGRAPH '95*, pages 219 – 228, 1995.

[17] C. Ou and Y. Ohno. Font generation algorithms for kanji characters. *Raster Imaging and Digital Typography*, pages 123 – 133, 1989.

[18] L. Ruggles. Letterform design systems. *Technical report STAN-CS-83-971*, 1983.

[19] T. Samoilov and G. Elber. Self-intersection elimination in metamorphosis of two-dimensional curves. *The Visual Computer*, 14(8/9):415 – 428, 1998.

[20] T. W. Sedergerg and S. R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH '86*, 20(4):151 – 160, 1986.

[21] A. Shamir and A. Rappoport. Extraction of typographic elements from outline representation of fonts. *EUROGRAPHICS '96*, 15(3):259 – 268, 1996.