Typeface Styling with Ramp Responses

Jörn Loviscach*
Fachhochschule Bielefeld (University of Applied Sciences)
Bielefeld, Germany

1 Introduction

Linear filters are standard tools of the artist in image, video and audio processing. This work demonstrates that they can also be used for *vector* graphics, in particular for typeface design. Serifs and other features of a typeface can be created and edited; gross effects are possible, too. In the software prototype, the user can freely define the shapes that the ⌞ type and the ⌟ type of convex axis-aligned rectangular corners should have after filtering. Corners of the types ⌝ and ⌜ will have the corresponding shapes, rotated by 180° degrees. The two target shapes can be edited as cubic Bézier paths. The result on a given text in a selected typeface is computed as vector graphics and displayed at an interactive rate.

In contrast to other recent approaches, the technique does *not* employ any high-level recognition of serifs or other features. Hence, it is fast and robust and produces consistent results for any shape that may occur in the font file.

2 Technique

Filtering planar curves by convolution is a tool long known in shape analysis [Mokhtarian and Mackworth 1992], in particular as a preprocessing step for pattern recognition. A curve is given by a mapping $\mathbf{r} : [0, s_{\text{total}}) \to \mathbb{R}^2$ of arc length—measured from an arbitrary reference point on the curve—to two-dimensional position. Here, s_{total} stands for the complete arc length of the curve. A curve specified in this manner can be filtered by convolution with a 2×2-matrix-valued function H:

$$s \mapsto \int_{-\infty}^{\infty} H(u)\, \mathbf{r}(s-u)\, du, \tag{1}$$

where the mapping $\mathbf{r}$ is interpreted as periodic with period s_{total}. The resulting curve will typically not be parameterized by arc length, but it is easy to check that it is closed, irrespective of the choice of H. Hence, the 2×2 element functions of the matrix H can be employed freely for design. Note that standard shape analysis does not use an arbitrary matrix function H, but employs a scalar Gaussian instead.

The relation of the four component functions of H to the visual result is not straightforward. Hence, the user interface cannot simply show these functions for editing. Interestingly, however, it turns out that specifying the target shape for the corners ⌞ and ⌟ completely defines these four functions and that any target shape for these corners can actually be achieved.

Key to this is the following observation: Let $s \mapsto (x_1(s), y_1(s))$ be the coordinates of the target shape for a ⌞-type corner and let $s \mapsto (x_2(s), y_2(s))$ be the coordinates of the target shape for a ⌟-type corner, both corners centered at a value s_0 of the arc length. (As opposed to the original curve, these target curves do not need to be parameterized by arc length.) Then a computation using Eq. 1 shows the following relations between the second derivatives of the coordinate functions and the elements of the matrix H:

$$\frac{d^2}{ds^2} \begin{pmatrix} x_1(s) & x_2(s) \\ y_1(s) & y_2(s) \end{pmatrix} = H(s - s_0) \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

*e-mail: joern.loviscach@fh-bielefeld.de

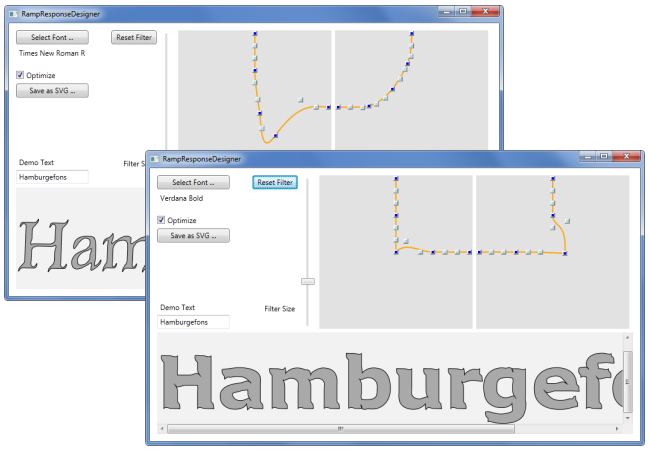

Figure 1: *The software prototype offers interactive controls for the target shapes of corners, demonstrated with two different settings applied to Times New Roman and Verdana Bold, respectively.*

This equation is uniquely solvable in terms of the matrix H. As the x and y components of the $\mathbf{r}$ mappings of axis-aligned corners are ramp functions, adjusting the corner shapes can be considered to be the design of ramp responses.

3 Implementation

The software prototype computes the filtered version of the $\mathbf{r}$ mappings for all contours of all glyphs of the entered text string in the selected font file. To this end, each contour is discretized into a polygon with hundreds or thousands of sides of constant length. While the user drags the control points of target shapes of the corners, the four functions of the matrix H are recomputed by numerical differentiation and applied to the shapes. For acceleration, the convolution process makes use of multi-threading by processing several glyphs in parallel. It turned out not to be necessary to speed up this process further, for instance by employing an FFT.

A slider controls the ratio of the arc length of the corner input fields to the arc length of the glyphs. This enables adjusting the overall size scale of the generated or affected features.

The software prototype is a quick tool for type effects and for sketching new typeface designs based on existing fonts. The result can be exported as an SVG vector graphics file. On export, the high-resolution polygonal shapes resulting from convolution can optionally be converted to a representation through cubic Bézier paths, with the obvious benefits for later editing, file size and geometric quality. With this final step, the method promises to be a helpful addition to graphics software and to font editing software.

References

MOKHTARIAN, F., AND MACKWORTH, A. K. 1992. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE TPAMI 14*, 8 (Aug), 789–805.