ELECTRONIC PUBLISHING, VOL. 6(3), 219–230 (SEPTEMBER 1993)

Dynamic regularisation of intelligent outline fonts

BEAT STAMM

Institute for Computer Systems
Swiss Federal Institute of Technology
CH–8092 Zürich, Switzerland

e-mail: stamm@inf.ethz.ch

SUMMARY

This paper introduces a novel way to perform dynamic regularisation of outline fonts. In the proposed font representation, the characters are decomposed into the components *glyph***,** *contour***,** *knot***, and** *number***. These components are scaled and mostly** *rounded before they are assembled***. Together with** *adroitly-defined Bézier curves***, this implies regularisation of the outlines without explicit grid-fitting, instructions, or hints. As a result, a single font representation permits font-scaling at increasing levels of detail, along with increasing type size and resolution.**

KEY WORDS Font representation Dynamic regularisation Medium and low resolution font-scaling

1 INTRODUCTION

Today's personal computers are equipped with raster displays and printers of highly varying resolutions. Modern typesetting programs are expected to deliver the best possible quality on all of these peripherals. For fonts intended for typesetting, this has two consequences. On the one hand, they have to be available for raster devices of any resolution and at any type size. On the other hand, they should display the highest possible level of detail at a given resolution and type size.

The first consequence is addressed by producing bitmapped fonts on demand from a generic representation. Fonts represented by their outlines, and scaled on-the-fly for screen resolution, are becoming the state of the art on today's personal computers [1]. The process of producing bitmapped fonts from outline fonts involves three main steps: *scaling*, *digitizing*, and *rendering*. Although the issues of digitizing (scan-converting) and rendering (outlining or filling) are not restricted to the topic of font-scaling, their correct and efficient implementation is a necessary prerequisite for it [2].

The second consequence is addressed by dynamic regularisation of the outlines. At screen resolution (typically 64 to 100 dpi), most details of 'artistic licence' inherently cannot be reproduced. Individual characters appear to be (and have to be) much more regular than their designer intended. At printer resolution (typically $\geq$ 300 dpi), more freedom of artistic expression can be represented in terms of pixels. To keep the digitizing and filling of characters as efficient as it can be, and as simple as it should be, the outlines are adapted to a given resolution and type size upon scaling. This process is called *dynamic regularisation*. It permits a single font representation to be used to produce bitmapped

CCC 0894–3982/93/030219–12
Received 1 September 1993
Revised 3 December 1993

fonts at increasing levels of detail, along with increasing type size and resolution, without a substantial loss of overall performance.

Starting with the fundamental raster problem and proceeding to the formalism for structuring fonts into components, Section 2 of this paper illuminates the background of our approach. Section 3 discusses and illustrates the topic of dynamic regularisation and leads to the demands on an appropriate font representation. Section 4 concludes with a summary, a brief comparison with existing approaches, and a suggestion in which direction to proceed with further research.

2 SOME BACKGROUND

2.1 The fundamental raster problem

Without loss of generality, we use integers for the coordinates in font space (i.e. the space in which we define the fonts). Let x denote such a coordinate. To scale x, we multiply it by a rational scaling factor $\sigma = n/d$, where both n and d are integers. With that, the product $\sigma \cdot x$ is a rational number. Now, in order to decide which pixels to turn on, the resulting scaling function $\mathrm{s}(x)$ eventually has to round back to an integer,

$$\mathrm{s}(x) := [\sigma \cdot x] \ ,$$

the rounding denoted by the brackets $[\ ]$. Given two coordinates x and y, and two rational numbers λ and μ denoting proportionality factors, this means that, usually,

$$\mathrm{s}(\lambda \cdot x + \mu \cdot y) \neq \lambda \cdot \mathrm{s}(x) + \mu \cdot \mathrm{s}(y)$$

that is, the scaling function is non-linear! As a consequence, naive font-scalers are likely to render equal stems unequally, since for such a stem often

$$\begin{aligned} \mathrm{s}(\texttt{rightEdge}) &\neq \mathrm{s}(\texttt{leftEdge}) + \mathrm{s}(\texttt{stemWidth}) \ , \ \text{although} \\ \texttt{rightEdge} &= \texttt{leftEdge} + \texttt{stemWidth} \ . \end{aligned}$$

Similarly, $\lambda = -1$ explains why symmetric serifs are likely to be rendered asymmetrically (that is, because $\mathrm{s}(\lambda \cdot x) \neq \lambda \cdot \mathrm{s}(x)$), and $0 < \lambda \ll 1$ gives an idea why tiny parts are prone to be dropped altogether (that is, because $\mathrm{s}(\lambda \cdot x) = 0$, although $\lambda \cdot \mathrm{s}(x) \neq 0$)). Since individual facets of the above inequality will strike us repeatedly, we call this inherent problem the *fundamental raster problem*. The fundamental raster problem explains the cause of many 'raster tragedies' (a term found in [3]). To avoid as many of these mishaps as possible when scaling fonts at low resolution, two conclusions are drawn next.

2.2 Conclusion 1: components

Equality and symmetry properties of characters are lost unless the fundamental raster problem is duly observed. Therefore, the characters have to be structured into (equal or symmetric) *components*. Essentially, this means that *glyphs* (or closed contours) and (open) *contours* are defined to be instances of *one and the same template*, rather than asking the rasterizing algorithm to render equal but independent copies of the glyphs *in the same way*. Unlike earlier approaches, which decomposed characters mainly into glyphs [4], the

fundamental raster problem dictates the use of *knots* (control or support points) and single *numbers* as components as well as glyphs. Knots and numbers can assume the role of yardsticks, whose instances are inserted repeatedly into the blueprints of characters, rather than requiring the respective distances to be scaled to the same lengths. The loss of linearity of the scaling function applies to all these levels of a character's blueprint.

2.3 Conclusion 2: the round-before-use rule

To assemble instances of components correctly, the components have to be rounded before they are used in the assembly. If they are not, they do not observe the fundamental raster problem. This is obvious for glyphs, for they are represented by a set of (rounded) pixels, and it should become obvious for numbers as well as soon as we understand numbers to represent vital dimensions of glyphs. Since this rule is of paramount importance, we have coined the term *round-before-use rule*. This conclusion is maybe a quite surprising one. Computer-science people tend to delay rounding as long as possible, so as to avoid rounding errors, and now we advocate the opposite!

2.4 A formalism for hierarchical outline fonts

We have defined a *formalism for hierarchical outline fonts* [5] to bridge the gap between the unstructured fonts we had and the structured fonts we needed. It is a declarative high-level language, because we would like to express what the font *is*, and not what the computer has to *do* in rendering the font. The entities to be declared in this language (*font*, *variant*, *character*, *glyph*, *contour*, *knot*, and *number*) are immutable and do not assume different values during font interpretation.

In this font language, the concept of *attributes* plays a central part. An attribute is a quality or a distinguishing feature looked upon as naturally or necessarily belonging to something. This is an important observation: with an attribute, we define what the component in question *is*, but not what the computer *might* have to know about it, nor what it *must* do with it. At the same time we would like to clearly distance ourselves from the popular terms *hint*, such as in [6], and *instruction*, such as in [7]. We understand a *hint* to be an indirect indication that suggests what *might* be the case. But the persistent vagueness of this term still permits some fact not to be the case, which does not go far enough. On the other hand, the imperative undertone of the term *instruction* does not quite blend in with the declarative nature of our formal approach. Specifying the order in which the operations *must* be performed by the computer goes too far. Notice, finally, that in the context of font-scaling we strongly believe the term *hint* to be often used in a misleading way — even in TrueType.

The most important attribute in this context is called the *hyperbolic scaling attribute*. This attribute is used for numbers denoting e.g. stem widths or serif heights. The attribute defines that the respective number persists under coarse scaling, while its omission permits the number to vanish in pixel space.

Two more attributes should be mentioned as well. One is used to override the round-before-use rule in cases where the latter is too restrictive. This is the case e.g. for numbers that denote the distances between on-curve and off-curve points in Bézier curves. Since these distances do not assume the role of reference distances and suchlike, they are irrelevant for the assembly of the glyph, and hence are not subject to the round-before-use rule. The

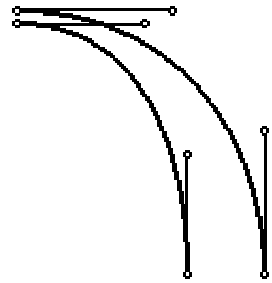

Figure 1. The first quadrant of a Times O*, defined with Bézier curves*

other attribute further specifies the nature of a contour. Such contour classes are e.g. polygons or Bézier curves, and others may be added [8]. In particular, extensibility permits smooth integration of answers to questions regarding the quality of rasterized curves. We definitely agree with [9] that dynamic regularisation cannot deal easily with such issues of the digital appearance of curvilinear glyphs.

3 IMPLIED DYNAMIC REGULARISATION

3.1 Curvilinear glyphs

Regularity of parts is only one aspect of font design. Quite often components that seem to be identical to the untrained observer, unveil subtle nuances on closer examination. Whatever the reason for such divergences from strict regularity may be, these details have to be respected by the font-scaler. Many of them cannot be represented in small type sizes at low resolution. Below a certain limit, a single unit in pixel space is far too coarse to express this or that tiny difference. Therefore the degree of regularity *must* increase along with decreasing resolution and type size. This transition to small type size at low resolution is the whole purpose of dynamic regularisation.

To get the upper hand of dynamic regularisation, we start with the mathematical representation of curvilinear glyphs. We have determined third-order Bézier curves to be the form most appropriate to the problem, because successive pairs of off-curve and on-curve points directly define the slope of the curve. This property can be exploited quite fruitfully if we see to it that the Bézier curves start and end at the locus of the local extrema of the desired shape (Figure 1).

With that, we obtain parallel tangents for pairs of Bézier curves. They will be used in subsection 3.1 to formulate optical corrections to the thicknesses of strokes, or to express the baseline overhangs and the mean-line and cap-line overshoots.

It is one thing to ask for something, but it is another to actually provide it. Inspired by [10] we implemented a program to do a spline-to-Bézier conversion, to bridge the gap between the natural spline curves we had and third-order Bézier curves defined in the way we wanted. For this conversion, analytic continuity at the on-curve points is not required; rather, geometric continuity is enough (Figure 2).

The outstanding advantage of all these efforts is that the dynamic regularisation of curvilinear glyphs has now been reduced to that of rectilinear lines. This is what we mean by adroitly-defined Bézier curves. Most control points now have a specific meaning as

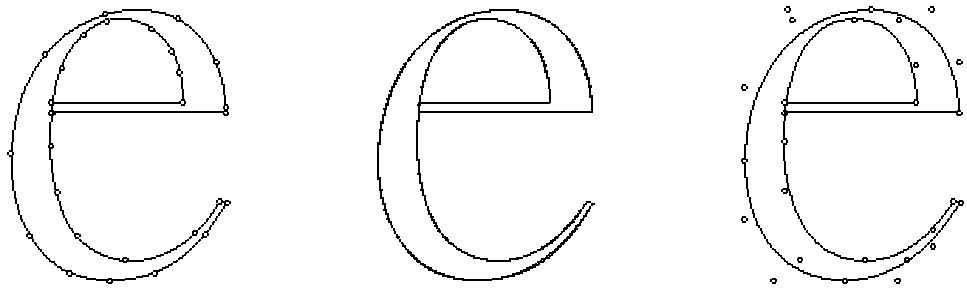

Figure 2. A Times e*, defined with natural splines (left) and with Bézier curves (right). The illustration in the middle depicts the degree of correspondence of the conversion program*

far as the shape of the entire glyph is concerned, which is in contrast to natural splines. As an immediate advantage, we point out that aligning the tangents with grid lines (*cf.* subsection 2.3) avoids extra and missing pixels at the local extrema of curves, as well as long flats, at no extra cost at all.

3.2 Optical corrections and snapping

In order to define the stroke thickness of curvilinear glyphs, the clever application of Bézier curves, together with the round-before-use rule, is considered next (Figure 3).

The illustration above shows that the thickest part of the curvilinear glyph is a little wider than the (regular) stem, while its thinnest part is just slightly narrower than the (regular) crossbar. These *optical corrections* to the stroke thickness of curvilinear glyphs are necessary if such glyphs are to give a well-balanced appearance in contrast to rectilinear glyphs. Yet they cannot be represented in small type sizes at low resolution.

Using the font language, the solution is to define components for the regular numbers `stemWidth` and `crossbarWidth`, which may be the same e.g. for all the capitals. These figures must not vanish under coarse scaling, hence they bear the hyperbolic scaling attribute. Then, the tangents of pairs of Bézier curves have their mutual distances (that is, the stroke thickness) depend on the sum of the regular value plus (or minus) an optical

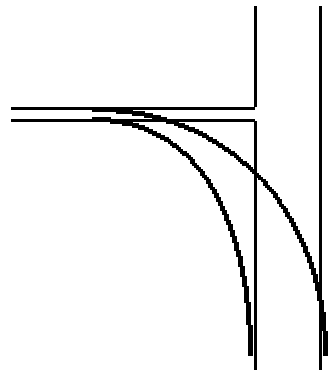

Figure 3. The first quadrant of a Times O*, overlaid by part of an* H

Figure 4. Times HO *at type sizes from 72 pt at 300 dpi down to 9 pt at 72 dpi, showing regularisation of the thicknesses of stroke and discrete dropping of reference-line overlaps*

correction. The correction may vanish under coarse scaling, hence it bears no special attribute for scaling. By dint of the round-before-use rule, finally, both the regular value and the optical correction are scaled and rounded before they are added to one another. Thus, both the two following equations hold:

$$\begin{aligned} \texttt{strokeThickness} &= \texttt{regularThickness} \pm \texttt{opticalCorrection} \\ \mathrm{s}(\texttt{strokeThickness}) &= \mathrm{s}(\texttt{regularThickness}) \pm \mathrm{s}(\texttt{opticalCorrection}) \end{aligned}$$

But since the optical correction is much smaller than the regular value, it will vanish for a sufficiently coarse scaling. As a result, the stroke thickness assumes the regular value without any further contribution! (At the same time, this ensures that the standard stems and crossbars will always come out regularly, without special stem-width control features and the like: the stems and crossbars simply refer to these numbers.)

The same scheme can be used to define the position of curvilinear glyphs relative to the horizontal reference lines (baseline, mean line and cap line). In order to give the eye the same impression of darkness that a rectilinear crossbar *on* the baseline would, the central part of a curvilinear glyph hangs slightly *below* the base line, while its ends remain *above* it. By symmetry, curvilinear glyphs not only overhang the baseline, but overshoot the mean line or the cap line as well. Therefore, the correct position of curvilinear glyphs explicitly includes in its definition the magnitudes of these overlaps. Under sufficiently coarse scaling this contour will drop its discrete overlaps and take its position *exactly* on the standard reference line, without special phase-control features and the like (Figure 4).

This is what an appropriate font representation should comprise. Even though it may seem somewhat elaborate, it closely mimics the way a font designer reasons about digital font design: designing, say, a Times O on screen, he does not mean to make the thickest vertical part of the character 36 units wide, but a few units wider than the regular vertical stems, whose widths happen to be 33 units. He would argue similarly about the thinnest horizontal part of the O, which he does not intend to be 6 units wide, but a single unit narrower than the regular crossbar of width 7 units. Analogous reasoning, finally, would lead him to the baseline overhangs or the mean-line and cap-line overshoots [11].

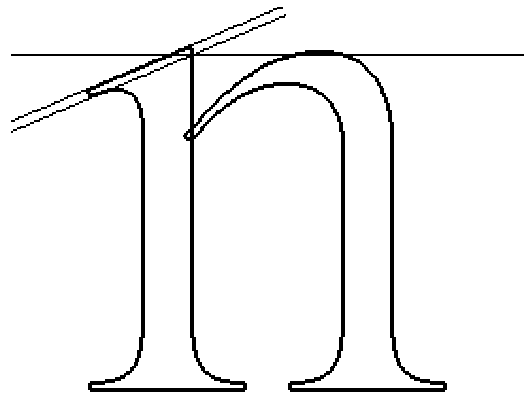

Figure 5. A Times n *in which the top edge of the head serif is inclined relative to the mean line*

In the previous example, discrete reference-line overlaps eventually *snapped* into a regular position. This idea can be used to align tilted or inclined straight lines as well.

As well as the mean-line overshoot of the arch of the n, Figure 5 shows several other features which are dynamically regularised. First, the inclined head serif ends slightly above the mean line. Clearly, this coordinate is formulated in terms of the mean line plus the respective optical correction. Next, the head serif starts well below the mean line. But this coordinate is expressed relative to the mean line as well. As a result, under sufficiently coarse scaling the top edge of the inclined head serif eventually snaps into alignment with the mean line. Lastly, the Bézier curve that makes up the bottom edge of the head serif is aligned with the top edge of the head serif. This is achieved by formulating the Bézier curve's tangent relative to the top edge, but asserting that it always maintains a minimum distance from the latter.

Notice that the half-serif width of the bottom serifs in Figure 6 eventually assumes the width of the stems, since it is defined relative to the stem width — this is achieved without special snapping values or half-serif-width or serif-thickness control features.

The examples in this section have in common *near misses* from *regular dimensions*. Let x denote such a regular dimension, and Δx a small part of it, then the fundamental raster problem explains why subtle nuances at high resolution may be enlarged disproportionately at low resolution:

$$\mathrm{s}(x + \Delta x) \neq \mathrm{s}(x) + \mathrm{s}(\Delta x) \ .$$

This is bound to happen whenever $\sigma \cdot x$ is just below a certain quantizing threshold, while $\sigma \cdot (x + \Delta x)$ is still above it. Mastering these *local phenomena* of *near-regularity* is merely

Figure 6. A Times n *at various type sizes from 72 pt at 300 dpi down to 9 pt at 72 dpi, showing dynamic regularisation of the stroke thickness and of different serifs*

Figure 7. Times Font*: 72 pt at 300 dpi (left) and 12 pt at 72 dpi (right)*

a matter of structuring a font into appropriate components, which in turn are to be scaled, rounded, and assembled in a straightforward way.

3.3 Caricaturing and balancing out

The spectrum of what dynamic regularisation should be able to achieve does not amount to merely 'ironing out' inconspicuous divergences. Rather, at the other end of the spectrum, dynamic regularisation has to cope with the requirements imposed by small type sizes at screen resolutions. At 72 dpi, the cap height of a 12 pt font is only just 8 pixels. For reasons of connectivity — if not to say, sheer decipherability — many regular dimensions must be severely exaggerated under coarse scaling.

At screen resolutions and the type sizes of plain text, different stroke thicknesses are drawn uniformly and divergent variants of serifs degenerate to a single pixel (Figure 7).

This *caricaturing* is a result of unconditionally safeguarding regular dimensions against vanishing, while seeing to it that delta dimensions *do* vanish under coarse scaling. For orthogonal rectangles, such as upright stems and horizontal crossbars, this is a mere question of proper font definition in terms of *number* components. Placing serifs next to a stem is understood to be like sticking small rectangles to the stem and sealing the joints that arise from the L and T-bars thus formed.

Under dynamic regularisation of the F in Figure 8, the stem and crossbar widths and the serif length are *balanced out* from a ratio of 33:7:4 to a ratio of 1:1:1. The more the resolution and type size decrease, the more these figures respond to hyperbolic scaling. Simultaneously, the seals of the serifs of the arm at the top, of the horizontal stroke in the middle, and of the foot at the bottom, seem to be 'soaked up' by their joints.

Matters are not that much different with glyphs of more general curvilinearity. The idea is to take a pair of concentric Bézier curves, as in Section 3, and define their tangents as if they were hooked into pairs of horizontally or vertically adjacent parallel guard-rails. Under dynamic regularisation, a tangent then can glide freely on the rail, but it cannot move away from it. Much as with ordinary railroads, a pair of rails introduces a gauge.

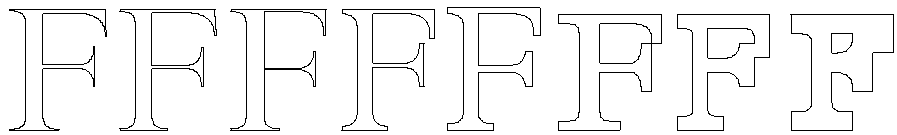

Figure 8. A Times F*, regularised for type sizes from 72 pt at 300 dpi down to 9 pt at 72 dpi*

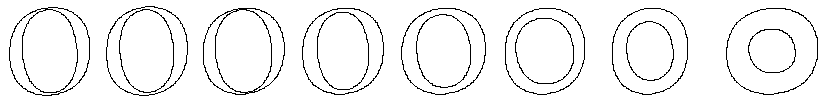

Figure 9. A Times o*, regularised for type sizes from 72 pt at 300 dpi down to 9 pt at 72 dpi*

This gauge corresponds to the maximum or the minimum stroke thickness, inclusive of optical corrections. Four such pairs of Bézier curves, that is, the topological equivalent of a full circle, are placed on an arrangement of rails that form two concentric orthogonal rectangles. These are the rectilinear convex hulls of the curves.

For the o in Figure 9, the outer one of these hulls extends from the baseline to the mean line (including overlaps). The inner hull moves further and further away from the outer one, along with decreasing type size and increasing exaggeration of the stroke thickness. An appropriate construct of the font language asserts that the control points glide on the guard-rails in appropriate proportions. With that, the dynamic regularisation of curvilinear glyphs has been reduced to that of orthogonal rectangles.

The o in Figure 9 had to undergo substantial deformations as a result of coarse scaling. But intuitively, this should be obvious: once it is clear that for a given type size and at a given resolution the vertical stroke of the o can best be represented by two units in pixel space, and once it is known that the horizontal stroke has to be drawn with one unit, then a continuous transition between them is a natural wish. This is precisely what is achieved when using third-order Bézier curves skilfully, as described here.

The examples in this section have in common an interplay of safeguarding characteristic dimensions and balancing out proportions. Let x denote such a characteristic dimension, and λ a portion thereof, then the fundamental raster problem explains why character decomposition should include *knots* and *numbers* as constituent parts. If it does not,

$$\mathrm{s}(\lambda{\cdot}x) \neq \lambda{\cdot}\mathrm{s}(x)$$

is bound to happen. Without a component x which is rounded before it is used, $\mathrm{s}(\lambda{\cdot}x)$ cannot refer to the result of $\mathrm{s}(x)$. Thus, mastering these *global phenomena* of *constrained proportionality* is a mere matter of structuring a font into components, which in turn are tagged with an attribute if they are to persist under coarse scaling.

3.4 An appropriate font representation

The best way of summing up the aspects of dynamic regularisation is to give a somewhat more comprehensive illustration.

Figure 10 shows a *family of curves*. Each member of this family is determined by its scaling factor, which by itself stems from the targeted resolution and type size. On sweeping through the scaling factors, individual characters are simplified from all their beauty down to a skeleton of topology — drawn at unit stroke thickness.

At the lower limit of resolution and type size, the requirement of a font definition is predominantly the possibility of expressing regularity, while at the upper limit, the whole gamut of ‘artistic license’ is made explicit. To recapitulate, a good font representation defines the following points:

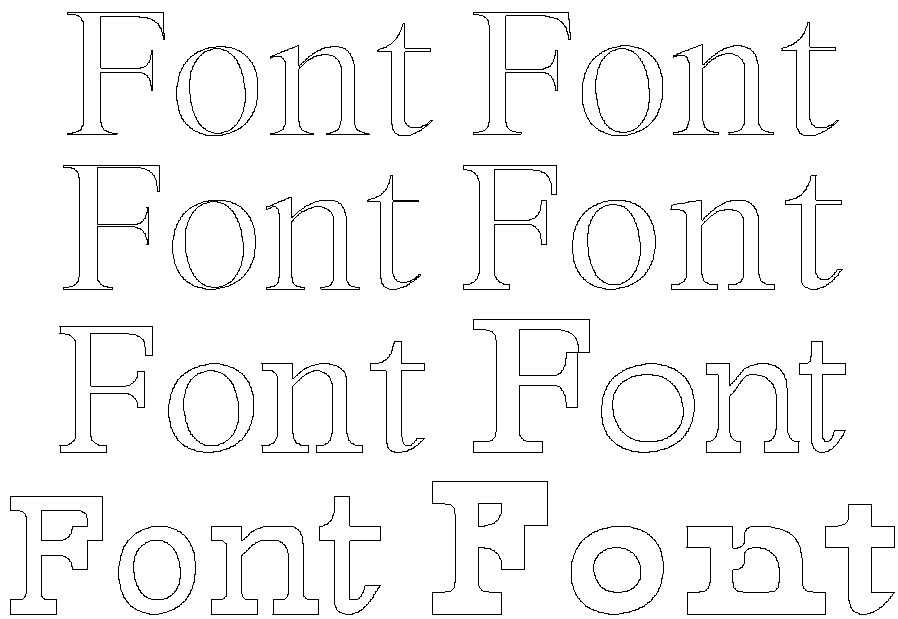

Figure 10. The outlines of Times Font *regularised for various type sizes from 72 pt at 300 dpi down to 9 pt at 72 dpi*

Regularity: The obvious consequence of the fundamental raster problem, the loss of regularity (equality, symmetry), is covered by decomposing characters into components (glyph, contour, knot, number) and safeguarding their existence (connectivity).

Near-regularity: Local aspects of dynamic regularisation (optical corrections, snapping) are taken care of by explicitly including in the characters' blueprints any divergence from strict regularity, and by rounding the blueprints' parts before they are assembled.

Constrained proportionality: Global aspects of dynamic regularisation (caricaturing, balancing out) are a consequence of safeguarding components against disappearance; they are realised by preserving proportions within the constraints imposed by the safeguards.

Together with the adroit application of third-order Bézier curves, this reduces to a font representation that provides for *well-behaved degeneration* in an amazingly natural way. Without resorting to a multitude of concepts and fancy features, dynamic regularisation is *implied* within this font representation. The appropriate font representation makes the intelligent outlines as simple as possible, but not any simpler.

4 CONCLUSIONS

In this paper we advocate a representation for *intelligent outline fonts*. The intelligence consists of representing the artistic aspects of font design explicitly. This particular representation allows to render fonts at increasing levels of detail along with increasing type size and resolution. At the same time, the process of digitizing and rendering a font on the fly becomes a relatively simple and straightforward task.

Our font representation, together with its interpretation, implies *dynamic regularisation* of outline fonts without any further contribution. This is in contrast to [7], which performs grid-fitting by explicit instruction. Thus, we might understand our approach as *implicit grid-fitting*. Together with varying scaling factors, our font representation defines a family of curves that interpolate different levels of detail of the outlines. This is related to the goals targeted by [12], which interpolates different weights, widths, sizes, and styles. Thus, we might understand well-behaved degeneration to be replacing the design axis *size*, or to be contributing a fifth design axis, labelled *detail*. Recently, [13,14] has introduced a simpler and technically more appealing approach that manages with a single master font. Compared to this approach, our font representation may be contributing to the axes labeled *expand* vs. *condense* and *optical scaling*.

The proposed font representation comprises a few simple concepts that are sufficiently close to the world of type design. These concepts are not related to idiosyncrasies specific to a particular rendering algorithm. With appropriate meta-information about topology and typography, as proposed in [15], we anticipate the feasibility of automated acquisition of complete outline font representations. We believe that this substantially eases the problem of making existing artwork usable on medium and low resolution devices.

ACKNOWLEDGEMENTS

Since the present paper has been realised as a part of my font-design project [16], I am particularly indebted to the project supervisor Prof. Dr. J. Gutknecht. The typographical foundations are due to the professional font designer H. Ed. Meier. Many thanks finally go to my colleague M. Hausner for his competent proof-reading and constructive criticism of the contents of this paper. Last but not the least, my special thanks go to R. Southall and J. André for their patient help with the final edition.

REFERENCES

1. Roger D. Hersch, 'Font rasterization: the state of the art', in *Visual and Technical Aspects of Type*, ed. R. D. Hersch, pp. 78–109. Cambridge University Press, (1993).
2. Jakob Gonczarowski, 'Fast generation of unfilled and filled outline characters', in *Raster Imaging and Digital Typography*, eds. J. André and R. D. Hersch, pp. 97–110. Cambridge University Press, (1989).
3. Peter Karow, 'Intelligent fontscaling', Technical report, URW Unternehmensberatung, Hamburg, Germany, (1987).
4. Philippe Coueignoux, *Generation of Roman printed fonts*, Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1975.
5. Beat Stamm, 'A formalism for hierarchical outline-fonts', in *Advanced Session of the IIEEJ*, Tokyo, Japan, (1992). Kogakuin University.
6. Adobe Systems Inc., *Adobe Type 1 Font Format*, Addison-Wesley, 1990.
7. Apple Computer Inc., *The TrueType Font Format Specification*, Cupertino, CA, 1990.
8. Beat Stamm, 'Object-orientation and extensibility in a font-scaler', *Electronic Publishing: Origination, Dissemination, and Design*, **6**(3), 159–170, (1993) (these proceedings).
9. Debra Adams and Richard Southall, 'Problems of font quality assessment', in *Raster Imaging and Digital Typography*, eds. J. André and R. D. Hersch, pp. 213–222. Cambridge University Press, (1989).
10. Jakob Gonczarowski, 'A fast approach to auto-tracing (with parametric cubics)', in *Raster Imaging and Digital Typography II*, eds. R. A. Morris and J. André, pp. 1–15. Cambridge University Press, (1991).

11. Hans Ed. Meier, 'Schriftgestaltung mit Hilfe des Computers', Technical report 167, Department of Computer Science, Swiss Federal Institute of Technology, Zürich, Switzerland, (1991).
12. Adobe Systems Inc., *Multiple Master Typefaces*, Mountain View, CA, 1991.
13. Peter Karow, 'Advanced typography', in *Advanced Session of the IIEEJ*, Tokyo, Japan, (1992). Kogakuin University.
14. URW Software & Type GmbH, Hamburg, Germany, *hz-Programm*, 1993. (See also, these proceedings, pages 283 sqq.)
15. Roger D. Hersch and Claude Bétrisey, 'Model-based matching and hinting of fonts', *Proceedings of SIGGRAPH'91, ACM Computer Graphics*, **24**, (1991).
16. Beat Stamm, *A hybrid approach to medium- and low-resolution font- scaling*, Ph.D. dissertation, Department of Computer Science, Swiss Federal Institute of Technology, Zürich, Switzerland, 1994 (to appear).