Next Generation Typeface Representations: Revisiting Parametric Fonts

Tamir Hassan
Pattern Recognition and Image Processing Group
Technische Universität Wien
Vienna, Austria
tam@tamirhassan.com

Changyuan Hu
20-20 Technologies Inc.,
Montreal,
QC, Canada
changyuan.hu@gmail.com

Roger D. Hersch
Peripheral Systems Laboratory,
Ecole Polytechnique Fédérale de Lausanne, Switzerland
rd.hersch@epfl.ch

ABSTRACT

Outline font technology has long been established as the standard way to represent typefaces, allowing characters to be represented independently of print size and resolution. Although outline font technologies are mature and produce results of sufficient quality for professional printing applications, they are inherently inflexible, which presents limitations in a number of document engineering applications. In the 1990s, the topic of finding a successor to outline fonts was a hot topic of research. Unfortunately, none of the methods developed at the time were successful in replacing outline font technology and this field of research has since then declined sharply in popularity.

In this paper, we revisit a parametric font format developed between 1995 and 2001 by Hu and Hersch, where characters are built up from connected shape components. We extend this representation and use it to synthesize several characters from the Frutiger typeface and alter their weights by setting the relevant parameters. These settings are automatically propagated to the other characters of the font family.

To conclude, we provide a discussion on next-generation font technologies in the light of today's Web-centric technologies and suggest applications that could greatly benefit from the use of flexible, parametric font representations.

Categories and Subject Descriptors: I.7.m [**Document and Text Processing**]: Miscellaneous; I.3.m [**Computer Graphics**]: Miscellaneous

General Terms: Experimentation

1. INTRODUCTION

Although the advent of outline fonts in the late 1980s was seen as a massive technological breakthrough, they do not present the ideal solution to representing character shapes. In order to represent an entire font family, it is necessary to create a separate font for each possible combination of attributes (e.g. bold, italic, condensed). Because these individual *styles* are represented independently of each other, it is not possible to select missing combinations of attributes or to continuously vary them.

Furthermore, there are several aspects of traditional typography which are either poorly or not at all supported by outline fonts, for example *optical scaling*. In the era of metal type, the shape of each *glyph* (character) varied according to its size. Smaller characters were usually slightly wider, in order to improve readability, and had thicker strokes in order to give the page a uniform appearance by keeping the font *colour* as constant as possible. The same applies to features such as small capitals, super- and subscript and mathematical formulae. In the move to digital outline fonts, such typographical subtleties as optical scaling have been lost and replaced by a straightforward mathematical transformation, producing inferior results.

In this paper, we revisit a previous parametric font representation (Chapter 3), extend it, and use it to synthesize several characters in different weights of the Frutiger typeface (Chapter 4). In Chapter 5 we describe further work necessary to develop the system to a mature stage and provide a discussion in Chapter 6 of the potential applications of the technology in the light of today's web-centric applications.

This work was funded in part by the Austrian *Akademisch-soziale Arbeitsgemeinschaft*.

DocEng2010, September 21–24, 2010, Manchester, United Kingdom.

2. RELATED APPROACHES

Multiple Master Fonts were introduced in 1991 from Adobe. A Multiple Master family consists of two or more basis outline fonts, known as *masters*, whose characters contain common control points. These control points allow the continuous modification of parameters or *design axes*. Unfortunately, Multiple Master fonts were never very popular and were discontinued.

A parametric font format that is still in use today is **Metafont**. Introduced in 1979 by Donald Knuth, it is nowadays mostly used in combination with the typesetting system TEX. Characters in Metafont fonts are built up using pen strokes defined by geometric equations, enabling several parameters to be varied continuously. More complex character shapes, such as those found in serif fonts, cannot be adequately represented using pen strokes alone. Therefore, the representation needs to be enhanced with outlines, which are then filled [1].

The **Infinifont** system [2] also builds character outlines using a sequence of operations, which may be adjusted using

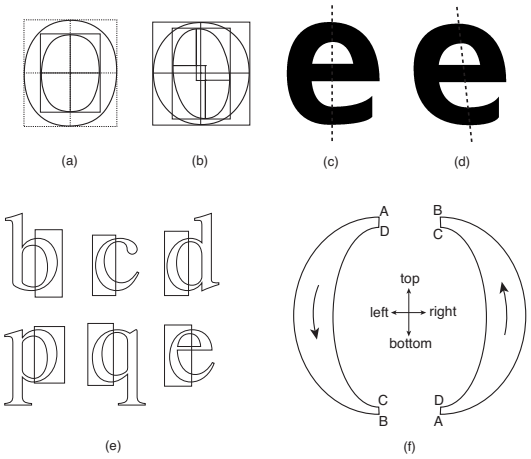

Figure 1: The loop and half-loop components

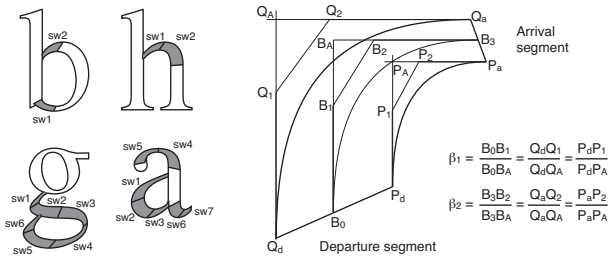

Figure 2: The sweep component

parameters. The system contains typeface-generic knowledge and can therefore approximate character shapes when only a small number of parameters are available.

In this paper, we concentrate on the **parametric representation of Hu and Hersch**, which was developed between 1995 and 2001 [3]. Here, characters are built up from their logical building blocks, such as strokes, curves, loops and serifs. Results of a first prototype, *CPFPage*, which synthesized the Times, Helvetica and Bodoni fonts, were published in 2001 [4].

3. THE CPFPAGE SYSTEM

In the representation from Hu and Hersch, characters are built up from the following basic component shapes:

- **strokes** to represent the horizontal and vertical stems, and other straight components of a character;
- **loops** and **half-loops** (see Fig. 1) to represent round parts of characters such as "o" and "b, d, p" respectively. Loops are defined by their inner and outer ellipses as well as an optional stress parameter η, which controls the relative orientations of the loop's inner and outer outlines;
- **sweeps** (see Fig. 2) to represent curved strokes. In contrast to (half-)loops, sweeps are defined by the orientations and widths of their departure and arrival segments, and by the Bézier curve of the central path. The letter "s" in sans-serif typefaces is often solely composed of sweeps;
- **serifs** to represent the terminals in serif typefaces;
- **path correction operators** to smooth the joins between certain components in serif faces.

For synthesizing sans-serif typefaces such as Frutiger, only the first three types of component primitives are usually required. A *character structure graph* can be used to represent how each of these components are joined together to create a character. Between different fonts, this structure graph can vary. Therefore, CPFPage includes several structure graphs for some characters, representing their most common *variations*. Please refer to [4] for a more detailed explanation of how characters are built from their components.

A font is defined by three parameter files. The *global* parameter file contains all the global measurements, such as x-height, stroke widths and character widths. The *local* parameter file contains parameters that affect the way only a single character is rendered. Whereas the shape of many characters, such as "h, m, n, u" can usually be accurately determined from the global parameters alone, other characters, such as "a", have a very individual appearance, and therefore contain a much larger number of parameters.

The third parameter file, the *group* parameter file, contains parameters which affect given groups of characters. For example, the curve-to-stem junctions of the characters "h, m, n, u" and "b, d, p, q" are identical in many typefaces and can therefore be adjusted uniformly.

4. A NEW FONT FAMILY

In this section, we describe our experiments in synthesizing the well-known Frutiger font family. We choose to concentrate on the first five letters of the lower-case alphabet, "abcde", as they contain a variety of different features, many of which also occur in other lower-case characters. Thus, while we were working on these five characters, other lower-case letters such as "o, p, q" automatically obtained their correct appearance.

In creating the first version of our parameterized Frutiger Medium (Fig. 3, Ib), it became clear to us that the CPFPage system had been designed primarily with serif typefaces in mind. The character "e" was modelled using a half-loop component, which resulted in the top and bottom curve centres having the same x-position (Fig. 1, c). As you can see from the original Type 1 font (Fig. 3, Ia), the centre of the bottom curve is shifted to the right. We found this to be a common feature of sans-serif fonts and not just limited to the Frutiger typeface.

We found that it was possible move the bottom curve towards the right by changing the loop stress parameter η. However, this had the undesired consequence of affecting the thickness of the top-left curve considerably (Fig. 1, d). We therefore created a new variation of the character "e" and replaced the half loop with two sweep components, allowing the bottom curve to be adjusted independently of the top curve (Fig. 3, Ic).

A further limitation that was observed was the inability to control stroke widths for each character individually. Bolder versions of sans-serif typefaces generally use slightly thinner strokes for "busier" characters such as "a" and "e". Already in the medium weight, these strokes were found to be visibly thinner. A new parameter was therefore added to adjust the stroke width in these characters.

Further structural changes were made to support the curved terminal at the bottom of the character "a" using an additional sweep component as well as to alter the thickness of the top terminal of the character "c" to match the bottom terminal. The improved version of our synthesized Frutiger

Medium, which now very closely resembles the original, is shown in Fig. 3 (Ic).

4.1 Parametric weight variation

Having achieved an accurate representation of Frutiger Medium, we used this as a starting point to generate the light and bold weights of the typeface, simply by modifying parameters. Fig. 3 (IIa) and (IIIa) show, as before, the original Type 1 rendition. Fig. 3 (IIb) and (IIIb) show the result that we obtained by modifying only the global parameters (see Table 1), i.e. character and stroke widths. Just by modifying half a dozen parameters, a new font style is generated whose quality is already acceptable for many purposes.

Fig. 3 (IIc) and (IIIc) show the result after modifying the remaining group and local parameters. In particular, moving certain control points of the "a" character achieves a better result. The curves of the characters "b" and "d", which were represented by sweep components, were also adjusted to look more natural. If an appropriate loop segment component had been available to represent these curves, perhaps this adjustment would have not been necessary.

As a further experiment, we tried to reconstruct the "Regular" weight by interpolating between the parameters of the "Light" and "Medium" weights. We first obtained what we believed to be the most influential parameter for boldness, the *vertical stem width*, from the Type 1 original. Based on this value, we performed a linear interpolation to calculate the remaining parameters. We were surprised by the result, as shown in Fig. 3, (IVb), which was almost perfect. For the "corrected" version (IVc), we made some slight adjustments to the character widths, which were evidently not in a linear relationship to vertical stroke thickness.

We also attempted to extrapolate the differences between the "Medium" and "Bold" weights to obtain the "Extra Black" weight of the typeface. The result, which is shown in Fig. 3 (Vb), was not of acceptable quality. It is worth noting that the shapes of extra bold fonts often differ considerably to the other weights, in order to produce an acceptable optical result within the limited available space. By reducing the character and horizontal curve widths (i.e. the parameters marked "*" in the rightmost column of Table 1), which had been over-increased by the extrapolation, we were able to obtain our first corrected result (Vc), which already represents a significant improvement. The second corrected result (Vd), which uses modified group and local parameters in addition, is even closer to the original font.

For our final experiment, we resynthesized Frutiger Medium Condensed based on our Frutiger Medium. The result is shown in Fig. 3 (VIb) which, apart from the obvious changes to condense the character widths, only required minor changes to the stroke widths. Although the weight of both faces was "medium", the stroke widths of the condensed version are slightly reduced to account for space limitations to produce a better optical result.

5. FUTURE WORK

As mentioned earlier, one possible improvement to the current representation would be to construct (half-)loops using sweep components, which might offer increased flexibility for character shape variation.

In our experiments with the Frutiger family, we found that many relationships between components were not present in the representation. For example, although the terminals of characters such as "a, c, e" are identical, this dependency was not present in the character structures.

The next step would be to analyse a greater number of typefaces and generate *groups of group parameters* for particular classes of font. It is a challenging task to find an appropriate way to specify this *knowledge* about letterforms, whilst still allowing the font designer to make any changes that he/she desires.

Once this information is adequately represented in the system, a further research direction would be the (semi-)automatic generation of parameterized fonts based from bitmap characters, which could contain noise or be incomplete.

Finally, a superior typeface representation will never become popular if there are no adequate interactive tools to create typefaces in this format, and a suitable GUI is therefore essential.

6. POTENTIAL APPLICATIONS

A key application of parametric fonts is in the conversion of documents from paper into electronic form. Even if the text has been correctly detected, the current trend of OCR software is to overlay a scanned bitmap image of the page on top of the text. The document therefore looks identical to the scanned image, but the hidden text remains selectable and searchable. This text is typically represented using a standard font whose metrics have been altered (and would therefore look very poor if it were visible).

Even with increasing resolutions, processing power and the use of antialiasing, it is still not a pleasurable experience to view a scanned document onscreen, and the printed result is also significantly worse than a digital original. Due to the increased file size, navigating such documents is slow, especially on low-power portable devices such as eBook readers and mobile phones.

Brailsford [5] recently described the difficulties in generating a vector representation of a scanned document. We believe that parametric fonts could play a very important role in this conversion, ultimately making it automatic. In many cases, such as the archiving operations of out-of-copyright material by **Google Books**, the fonts do not exist in digital form. Even when they are available, they may have significantly different metrics. Furthermore, each new use of a commercial font would require obtaining the relevant licences. The use of parametric fonts to *approximate* the fonts used in scanned materials would sidestep these licensing and availability issues, and still create a much better, scalable visual result than the scanned image of a page.

The further potential applications of this technology are perhaps more obvious, including font compression, automatic detailed classification (the format is *self-describing*) and the creation of new styles or characters (e.g. symbols, logos, Cyrillic characters, etc.) for an existing typeface with little additional effort.

7. REFERENCES

[1] D. Knuth: Lessons learned from Metafont. In *Digital Typography,* pp.315–338, CSLI Publications, Stanford, 1999.

[2] C.D. McQueen III and R.G. Beausoleil: Infinifont: a parametric font generation system In *Elec. Pub.* 6(3):117–132, 1993.

[3] C. Hu: *Synthesis of parametrisable fonts by shape components.* PhD thesis, EPF Lausanne, 1998.

[4] C. Hu and R.D. Hersch: Parameterizable fonts based on shape components. In *IEEE Comp. Graph. Appl.* 21(3):70–85, 2001.

[5] D.F. Brailsford: Automated re-typesetting, indexing and content enhancement for scanned marriage registers. In *Proceedings of DocEng '09,* pp. 29–38, 2009.

	Light	Interpolated Regular	Corrected Regular	Medium	Med. Cond.	Bold	Extrapolated Extra Black	Corrected Extra Black
Global parameters								
Vertical stem width	64	90	90	130	124	172	248	248
Vertical curve width	64	90	96	131	120	181	271	250*
Round letter width	453	482	505	526	460	588	700	653*
Stem–curve width	358	359	378	361	300	386	431	365*
Narrow hor. stem width	54	67		87	75	102	129	107*
Horizontal curve width	54	72		100	93	123	165	143*
X-height	492	494		496	493	501	510	502
Group parameters								
"e"-bar position	0.51	0.51		0.51		0.5	0.48	0.49
Loop centre x-pos. external	0.53	0.55		0.58		0.67	0.83	0.76
"b" lower sweep A	1.34	1.36		1.39		1.42	1.47	
"b" upper sweep A	0.90	0.89		0.88		0.90	0.94	
Local parameters "a"								
Main width	0.69	0.65		0.60		0.58	0.54	
Secondary width	0.64	0.63		0.61		0.61	0.61	0.63
Dot angle	88	88		88		88	88	90
Head width	0.9	0.9		0.9		0.75	0.48	0.6
Belly left-most height	0.27	0.27		0.27		0.29	0.33	0.29
s6 curve A.x	1.12	1.12		1.12		1.28	1.57	1.31
Belly bottom-most x	0.55	0.55		0.55		0.56	0.58	
s8 arrival y	0.38	0.38		0.38		0.49	0.69	0.52
Vertical stem width %	0.97	0.95		0.92		0.92	0.92	
Protrusion of bottom terminal	0.02	0.02		0.03		0.03	0.03	
Width of bottom terminal	0.95	0.95		0.95		0.98	1.03	
Height of bottom terminal	0.15	0.23		0.35		0.2	-0.07**	0.00
Local parameters "c"								
Main width	0.79	0.76		0.71		0.75	0.82	0.76
Secondary width	0.79	0.76		0.72		0.74	0.78	0.76
Tail width	0.80	0.80		0.80		0.70	0.52	
Loop centre x-pos. external	0.70	0.70		0.69		0.69	0.69	
Loop centre x-pos. internal	0.74	0.74		0.75		0.75	0.75	
Dot centre y	0.98	0.98		0.98		0.97	0.95	
Tail y	0.01	0.01		0.01		0.02	0.04	
Dot angle	85	85		85		85	85	90
Tail angle	86	86		86		86	86	90
Local parameters "e"								
Main width	0.82	0.82		0.82		0.81	0.79	0.83
Tail width	0.80	0.80		0.80		0.70	0.52	
Wide miniscule vertical curve width	1.00	0.97		0.93		0.93	0.93	
Wide miniscule horizontal curve width	1.00	0.97		0.93		0.93	0.93	0.88
Width of right curve %	1.00	1.00		0.99		0.90	0.74	0.87
Loop centre x-pos. internal	0.50	0.50		0.50		0.50	0.50	0.52

* For the first corrected version of this font, only the parameters marked with an asterisk were modified from the extrapolated values.
** As this negative value did not fall within the range of accepted values for this parameter, it was set to zero.

Table 1: Parameters that were modified in the various styles of the Frutiger typeface. Please note: Blank cells mean that the parameter was not modified from its previous value (the neighbouring cell to the left)

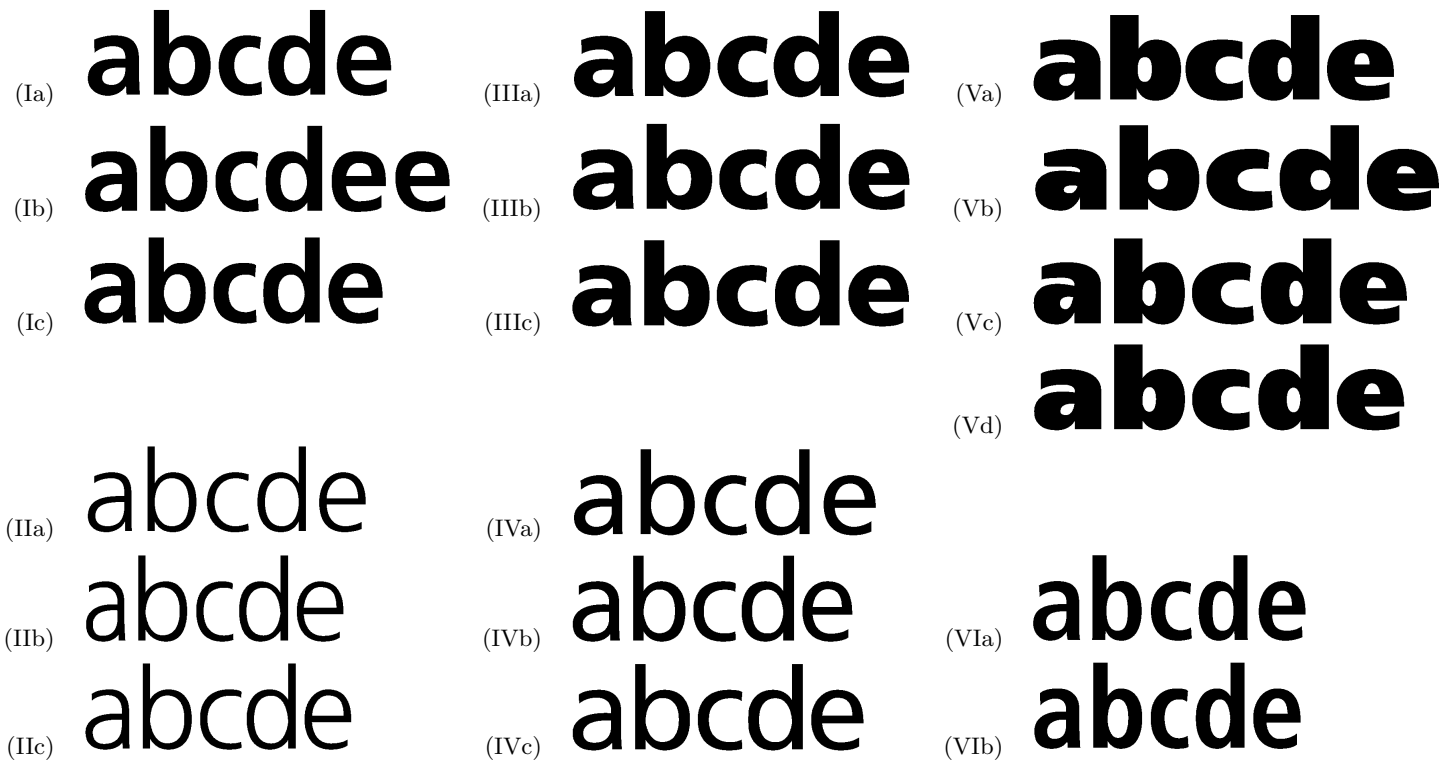

Figure 3: Synthesis of the various versions and styles of the parametric Frutiger typeface. Figures with the suffix "a" represent the original Type 1 font, "b" the initial synthesized version, and subsequent letters relate to further improvements, as detailed in the referring text