Automatic Synthesis of Contrast Controlled Grayscale Characters with Component-Based Parametrisable Fonts

Changyuan Hu, Roger D. Hersch
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
RD.Hersch@epfl.ch

Abstract

A few years ago, a *perceptually-tuned grayscale character generation* technique was developed in order to automatically synthesize grayscale characters looking like manually-tuned pixmap characters. Weight and contrast controlled grayscale characters are obtained by grid-fitting the scaled character contours. However, this technique requires that hinting information be added to the outline font description. Adding hinting information to each outline character generally requires a considerable amount of human intervention.

Our *component-based parametrisable font system* is a newly developed font description and reproduction technology. It incorporates for each basic character shape a software method responsible for the synthesis of an instance of that character. A given font is synthesized by providing appropriate font parameters to these character synthesis methods. Numerous concrete fonts can be derived by simply varying the parameters. Such variations offer high flexibility for synthesizing derived fonts (variations in condensation, weight and contrast) and enable saving a considerable amount of storage space. This paper shows that with component-based parametrisable fonts, high quality perceptually-tuned grayscale characters can be generated without requiring hinting information. Generating perceptually-tuned grayscale characters with parametrized component-based fonts consists in automatically adapting the phase of some of the character's parameters in respect to the underlying grid and in ensuring that thin character parts are strong enough not to disappear (weight-control).

1. Component-based parametrizable fonts

Most commercial font systems use outline fonts for the digital presentation of typefaces. Outline fonts are well suited for character rasterization and printing. They however only incorporate implicit information about important font features such as the width of stems, bars and bowls, the distance between vertical stems and bowls and the parameters determining junctions between character elements and determining the shape of terminal elements (serifs). Efforts are made to create a more flexible character representation incorporating explicit information about features of the character, thus enabling the synthesis of coherent font variations, for example by varying weight and contrast. The main idea is to describe characters by an assembly of appropriate structure elements (components). Several attempts have been made to describe characters by structure elements. D. Knuth described his Computer Modern fonts by horizontal, vertical and diagonal strokes as well as by round parts [5]. These strokes and round parts are given by sequences of pen positions and orientations. More recently, Bauermeister et al. created the Infinifont system enabling resynthesizing an existing font from universal font generation rules and from parametric data specific to that font [6][9]. Our own efforts [11] aimed at defining character components suitable for the generation of existing and of derived typographic fonts, incorporating variations in condensation, weight and contrast.

Fig. 1 shows a character assembled from structure elements, i.e. the two vertical stems are synthesized by the component named *stem*, the round arch is synthesized by two connecting *sweep* components, and the terminals are synthesized respectively by the *serif* and *top-serif* components. For the round character structure elements which are similar to a whole or a half of letter "o", we synthesize them by the *loop* and respectively the *half-loop* components (letter "o" and "b" in Fig. 2).

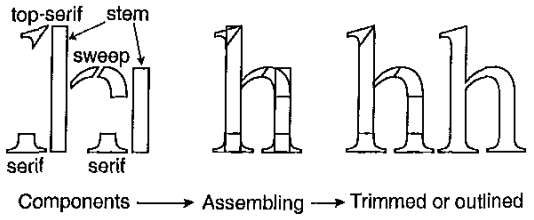


Figure 1. Building a character with components.

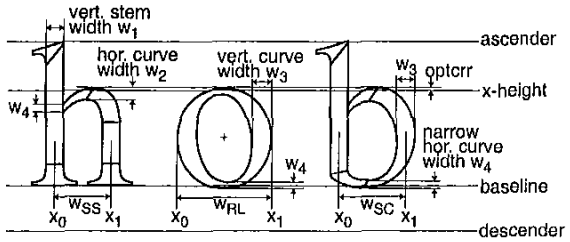


Figure 2. Parameters of typical component-based characters.

In our component-based font parametrization method, the shape of a character is synthesized by synthesizing the shape of each component from font parameters. Font parameters have clearly defined typographical meanings (Fig. 2). They are used to describe the characters' typographical features, such as the character height alignment lines (reference lines), the widths of stems and arches, the metrics of serifs, the angles of junctions or

0-7803-5467-2/99/$10.00 © 1999 IEEE

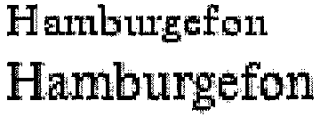
slant serifs, etc. Some parameters are globally available and are applied to all or to a group of several characters to ensure the coherence of a font, respectively the coherence of a group of parameters. Some parameters have only a local meaning for one specific character. The concrete values of parameters may be the distance between two points, the intermediate position (proportion) of one point between two other points, the angle of a junction or of a slant serif, etc.

Components are assembled according to a dependency order: the main components, such as the two vertical stems in letter "h" or the left stem and the right half-loop of letter "b", are first placed according to a parameter controlling their horizontal distance. The position of other components such as connecting sweeps and serifs are determined by their relationship to the previously placed main components. Such a construction of characters by components enables connection relationships between components to be maintained when modifying parameter values. Derived characters of varying condenstation, weight and contrast can therefore be obtained by modifying a suitable subset of parameters [12].

2. Perceptually-tuned grayscale fonts

In order to improve the display of text on limited resolution computer displays, researchers have tried to trade-off the lack of spatial resolution for an increased number of intensity levels [1, 2, 3]. Especially with respect to typographic text display on limited resolution CRT and LCD display devices, it has been shown that using grayscale makes font dependent character features visible which disappear when displaying text with bilevel characters. Though they can more accurately render font differences, grayscale characters generated by filtering and resampling high-resolution bilevel master characters look rather fuzzy. Such characters do not have a high enough contrast along their horizontal and vertical edges and thin character parts tend to disappear. In addition, similar character parts (bars, stems, etc..) tend to look dissimilar.

a) Box filter



b) Manually edited

Figure 3. Characters generated by filtering and resampling compared with manually edited grayscale characters

Different instances of similar character structure elements such as vertical bars or curved stems look dissimilar because they do not have similar intensity profiles (Fig. 3a). This is due to the fact that the original bilevel master character incorporates frequencies well beyond the Nyquist limit (1/2 the resampling frequency) and that LCD displays are capable of displaying individual pixels as constant intensity squares of unit size, thereby enabling high intensity gradients between neighbouring pixels.

The phase of character elements with respect to the target pixel grid has a strong impact on their intensity profiles, especially at small sizes and low resolution (Fig. 4a).

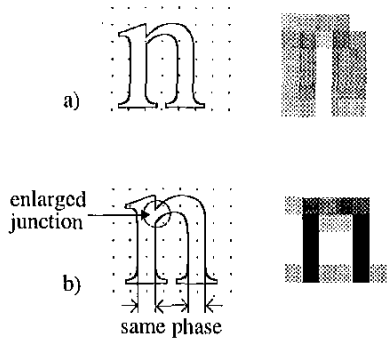


Figure 4. The phase of the vertical bars defines their intensity profile: (a) without outline phase control and (b) with outline phase control

Creating optimal grayscale characters involves a manual pixel-by-pixel design that must follow strict typographic rules. In order to automatically generate improved quality grayscale characters similar to the characters which would be designed manually by skilled typographic designers, the so-called *perceptually-tuned grayscale font generation method* [10,13] successfully applies typographic rules for the automatic grayscale character generation process. Typographic rules derived from manual pixel-by-pixel design have been translated into a set of character outline grid-fitting rules, such as phase-control of the vertical main stems in order to enhance the sharpness of their left edge, thickening thin strokes when their width is too small, and phase-control of round strokes so that the extrema of round character (such as "c", "e" and "o") have a coherent appearance.

To fulfil these grid-fitting tasks, traditional outline font technology requires additional information called hints (or instructions in TrueType terminology) enabling the rasterizer to adapt the scaled character contour to the pixel grid by applying slight modification to the character shape contours [10].

3. Synthesis of perceptually-tuned grayscale characters with component-based parametrisable fonts

With the component-based parametrizable font technology, grid-fitting can be applied without explicit hinting information. This is possible, because the component-based description of a character explicitly specifies the typographical meaning of each character component, e.g. the *stem*, the *serif*, the *loop* and the connecting *sweep* components. The component synthesizer knows how to synthesize the shape of a component from font parameters. Our approach consists in modifying global and local parameters (Fig. 2) which control the phase of critical component contours in respect to the grid.

We consider the placement and rasterization of component-based characters on a 2D pixel grid where pixels have a size of one and where pixel centers have integer coordinates. The scaling factor *scaleFac* specifies how much the original global parameters need to be scaled to yield the final character at the target size.

We control the phase of reference lines in order to ensure that the grayscale representation of round letters (such as letter "o") remains vertically symmetric and that foot serifs remain sufficiently visible. Reference lines should therefore be placed at the boundaries between rows of pixels. Let us first place the baseline at the nearest boundary between pixels.

$$baseline^{new} = \lfloor baseline \times scaleFac \rfloor + 0.5$$

Other reference lines are placed relatively by rounding the distances between them and the baseline. Therefore, the other reference lines are also located on pixel boundaries.

$$d_1 = \lfloor (xheight - baseline) \times scaleFac + 0.5 \rfloor$$

$$xheight^{new} = baseline^{new} + d_1$$

$$d_2 = \lfloor (ascender - baseline) \times scaleFac + 0.5 \rfloor$$

$$ascender^{new} = baseline^{new} + d_2$$

$$d_3 = \lfloor (descender - baseline) \times scaleFac + 0.5 \rfloor$$

$$descender^{new} = baseline^{new} + d_3$$

We align the left edge of all vertical stems to horizontal pixel boundary by first placing the centerline of one of the vertical stems in a character (for example the left ascender stem of letter "h") to a phase ensuring that the left side of the stem is on a pixel boundary. Parameter w_1 defines the vertical stem width (Figure 2). The position of the second stem is calculated by adding to the position of the first stem the rounded scaled value of the *stem-to-stem distance* (parameters w_{SS} in Fig. 5).

$$w_{SS}^{new} = \lfloor w_{SS} \times scaleFac + 0.5 \rfloor$$

$$x_0^{new} = \left\lfloor \left(x_0 - \frac{w_1}{2} \right) \times scaleFac \right\rfloor + 0.5 + \frac{w_1 \times scaleFac}{2}$$

$$x_1^{new} = x_0^{new} + w_{SS}^{new}$$

For round letters, such as letter "o", "c" and "e", we place the horizontal position of the center of symmetry on a pixel center so as to ensure that left half and the right half of the letter are symmetric in respect to the underlying grid and are therefore rendered with a symmetric intensity profile. The *round letter width* parameter (w_{RL} in Fir. 2) needs only a simple scaling:

$$w_{RL}^{new} = w_{RL} \times scaleFac$$

$$x_0^{new} = \lfloor x_0 \times scaleFac \rfloor - \left(\frac{w_{RL}^{new}}{2} - \left\lfloor \frac{w_{RL}^{new}}{2} \right\rfloor \right)$$

$$x_1^{new} = x_0^{new} + w_{RL}^{new}$$

For letters with one vertical stem and a half loop, such as "b", "d", "p" and "q", we control the phase of the vertical stem in the same way as for letter "h" and scale the distance between the stem and the half loop (w_{SC} in Fig. 2). No phase control is applied to the half-loop component.

Finally, the width (thickness) of a thin curved or a diagonal stroke needs to be increased if it is too thin, i.e. less than 1 pixel. This can be done by enforcing a lower limit (for example 0.75) to the corresponding width parameters (w_4 in Fig. 2) after scaling.

$$w_4^{new} = max(0.75, w_4 \times scaleFac)$$

Fig. 5a illustrates the synthesized component-based characters obtained by applying the previously described parameter scaling, phase control and thickness modification rules.

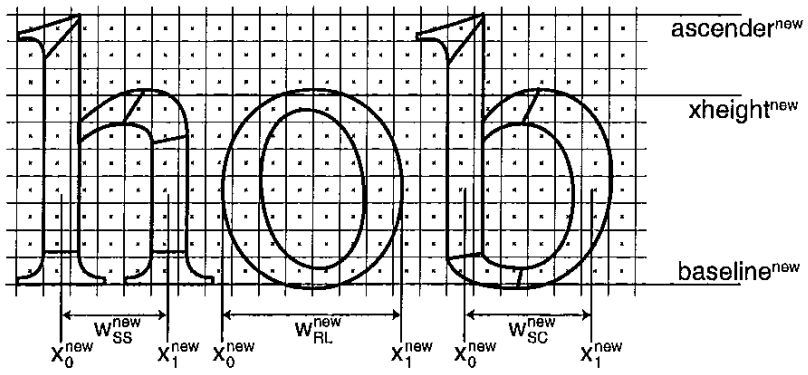


Figure 5. The contours of synthesized component-based characters are adapted to the pixel grid by applying the on-the-fly parameter scaling and modification rules.

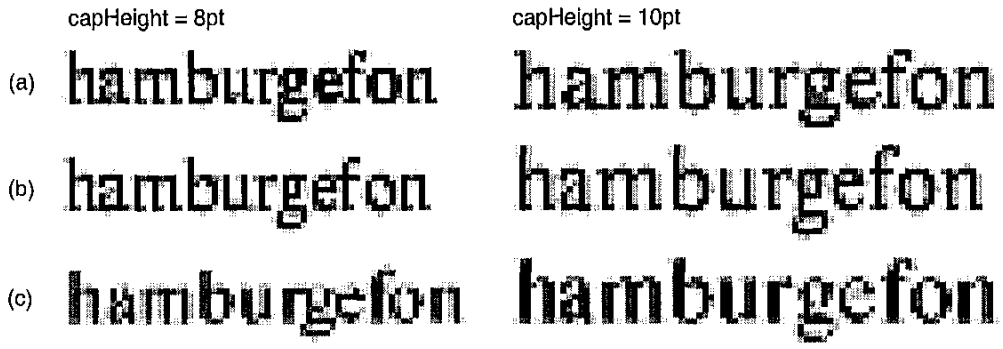


Figure 6. (a) Component-based generation of grayscale characters, with improved contrast obtained thanks to phase control of stems and round parts and to thickness control of thin strokes; (b) Perceptually-tuned grayscale characters generated by applying hinting instructions to the grid-fitting process; (c) corresponding grayscale characters generated by filtering and resampling (traditional technique).

Fig. 6a shows the enlarged image of the grayscale characters generated by our component-based parametrizable character synthesis system [11], in comparison with the perceptually-tuned grayscale characters (Fig. 6b) generated by the previously developed hint-based grid-fitting system [10] and in comparison with traditional grayscaling, i.e. filtering and resampling of high resolution master character (Fig. 6c).

4. Conclusions

The component-based parametrizable font system enables the synthesis of coherent fonts. Paraneters such as the relative position of reference lines, the spacing between main character parts, the bar and stem widths define to a large extent the shapes of the synthesized characters. By controling the phase of these parameters, sufficient grid-fitting is achieved to generate high-contrast uniformely looking grayscale characters. Phase control is applied to reference lines, to stem and bar centerlines and to width parameters. At small sizes, a sufficiently strong structure of individual strokes is achieved by ensuring a minimal size of stem and of curved stroke width parameters (weight control). Phase and weight control are completely independent of component-based character generation: a separate program is responsible for phase and weight control of specific fonts paramters. After phase and weight control of parameters, component based bilevel characters are generated at a resolution 4x4 times higher than the target resolution. Each 4x4 bitmap square is in phase with one grayscale pixel and yields one grayscale pixel whose intensity is one of 17 possible intensities.

Once characters are described by components, the presented method is simple to implement and does not require, besides component contours, any additional information. It may become part of the facilities offered by a completely new generation of component-based font synthesizing systems. Its well contrasted and uniformly looking grayscale characters on middle and low resolution displays, especially on LCD displays, may considerably improve reading confort when accessing electronic documents (PDF files, Web pages).

5. References

[1] F. Crow, "The Use of Grayscale for Improved Raster Display of Vectors and Characters", Computer Graphics (Proc. Siggraph'78), 12(3), 1978, 1-6.

[2] J. Warnock, "The Display of Characters Using Gray Level Sample Arrays", Computer Graphics (Proc. Siggraph'80), 14(3), 1980, 302-307.

[3] A. Naiman, A. Fournier, "Rectangular Convolution for Fast Filtering of Characters", Computer Graphics (Proc. Siggraph'87), 21(4), 1987, 233-242.

[4] P. Karow, "Automatic Hinting for Intelligent Font Scaling", *Proc. Raster Imaging and Digital Typography 89, (Eds. J. André, R.D. Hersch)*, Cambridge Univ. Press, 1989, 232-241.

[5] D. E. Knuth, *The METAFONT book*, Addison-Welsey, Reading Mass. 1986.

[6] C. D. McQueen and R. G. Beausoleil, "Infinifont: a parametric font generation system", *RIDT 94, Electronic Publishing*, Vol.6(3), John Wiley & Sons, 1993, 117-132.

[7] R. D. Hersch, C. Bétrisey, "Model-based matching and hinting of fonts", *Proceedings SIGGRAPH'91, ACM Computer Graphics*, Vol. 25, No. 4, 1991, 71-80.

[8] A. Shamir, A. Rappoport, "Extraction of Typographic Elements from Outline Representations of Fonts", *Proc. Eurographics 1986, Computer Graphics Forum*, Vol. 15, No. 3, 259-268.

[9] Bauermeister et al., "Method and system for creating, specifying, and generating parametric fonts", *United States Patent No. 5586241*, Dec. 17, 1996.

[10] R. D. Hersch, Claude Betrisey and Justin Bur, "Perceptually Tuned Generation of Grayscale Fonts", *IEEE Computer Graphics and Applications*, Vol. 15, No. 6, 1995, 78-89.

[11] C. Hu, "Synthesis of Parametrisable Fonts by Shape Components", *Ph.D thesis N° 1905*, École Polytechnique Fédérale de Lausanne, 1998.

[12] J. Hertz, C. Hu, J. Gonczarowski, R. D. Hersch, "A Window-Based Method for Automatic Typographic Parameter Extraction", *Proc. EP'98/RIDT'98*, Eds. R. D. Hersch etc, LNCS 1375, Springer 1998, 44-54.

[13] K. O'Regan, N. Bismuth and R. D. Hersch, "Legibility of Perceptually-Tuned Grayscale Fonts", *Proc. ICIP-96*, Ed. P. Delogne, Vol 1, 1996, 537-540.