Model-based Matching and Hinting of Fonts

Roger D. Hersch, Claude Betrisey

Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland

Abstract

In today's digital computers, phototypesetters and printers, typographic fonts are mainly given by their outline descriptions. Outline descriptions alone do not provide any information about character parts like stems, serifs, shoulders, and bowls. But, in order to produce the best looking characters at a given size on a specific printer, non-linear operations must be applied to parts of the character shape. At low-resolution, grid-fitting of character outlines is required for generating nice and regular raster characters. For this reason, grid-fitting rules called hints are added to the character description. Grid-fitting rules require as parameters certain characteristic points within the shape outlines. In order to be able to detect these characteristic points in any given input font, a topological model representing the essence of the shapes found in typographic latin typefaces is proposed. This model includes sufficient information for matching existing non-fancy outline fonts to the model description. For automatic hint generation, a table of applicable hints is added into the topological model description. After matching a given input shape to the model, hints which can be applied to the shape of the given font are taken and added to its outline description. Furthermore, a structural description of individual letter shape parts using characteristic model points can be added to the model. Such a description provides knowledge about typographic structure elements like stems, serifs and bowls.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling.

Additional Key Words and Phrases: Digital typography, outline fonts, topological model, shape matching, grid-fitting, automatic hinting.

1 Introduction

In the last century, hundreds of new font families have been created for metal type and photocomposition. Photocomposition equipment manufacturers and font trading companies now offer thousands of different fonts. Digital fonts are described by their outlines, mainly in the form of straight line and spline segments.

Character outline descriptions however are by no means adequate for creating and manipulating typographic shapes. They are only suitable for generating scalable fonts on high-resolution photocomposers. Outline characters should be considered only as the result of the font design process. During type design, type creators mainly consider important font features like stoke width relationships, curvature of rounded shapes, serif thickness and length, junctions between serifs and stems, alternation of white and black space, etc.. Previous work in digital typography has shown that typefaces can be assembled by using structural shape parts incorporating the basic font features [6] [13]. Computer-aided type design [9] starts with the study of structural letter parts like serifs, arms, shoulders, bowls, junctions and terminal endings (figure 1) and the generation of variants of representative characters (E, H, O, n, b, o, a, e, i). The remaining letters, numbers and signs of an alphabet can be partly synthesized into outline characters by assembling these structural letter parts. Previous attempts to develop computer tools for converting automatically structural descriptions into outline descriptions and vice-versa produced only limited results [1].

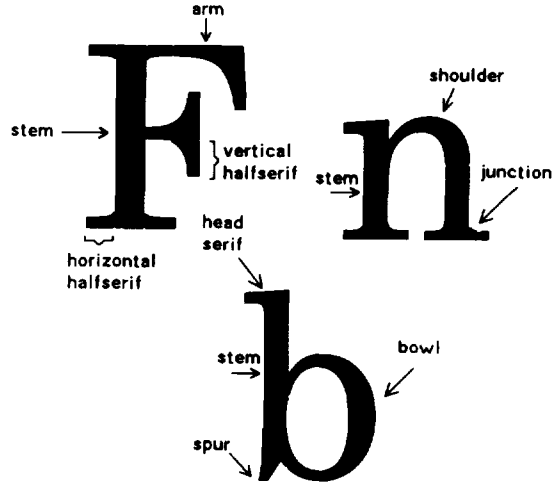


Figure 1: Structural letter parts [10]

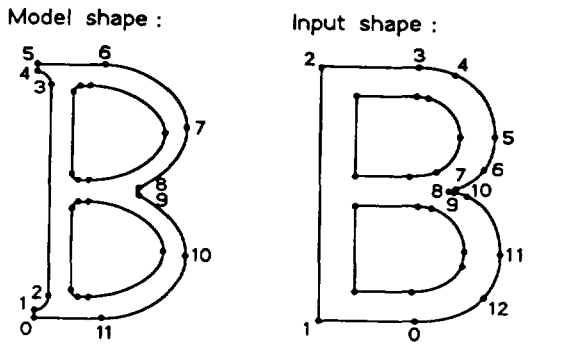
Font outline descriptions are not sufficient for rendering outline fonts on middle and low-resolution devices like 300 dpi printers and 100 dpi displays. Recently, techniques have been developed for the grid-fitting of character outlines to a given rasterization grid [11] [2] [12]. These techniques require additional information in the form of rules specifying the kind of constraints to be applied on certain character parts. These rules, called *hints* or *grid constraints* refer to selected outline support points defining the width or thickness of particular character parts like staight stems, bowls, serifs and diagonals. The application of grid constraints adapts the original outline shape to the grid in order to maintain symmetry and regularity in the produced discrete shape. For a given font size, the rasterized characters will have equal stem width, identical serifs and nice discrete arcs.

Manually incorporating hints into outline font descriptions is tedious and labor intensive. Due to the large amount of available outline

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

©1991
ACM-0-89791-436-8/91/007/0071 $00.75

fonts, automated creation of hinted outline fonts is a necessity. For the application of basic hints on stems, bowls and shoulders, automatic hinting can be based on the recognition of horizontal and vertical bars and on the recognition of curvilinear shape extremas [3]. For the generation of more accurate hints like the control of serifs, of diagonal bars and the control of uniform weight at small sizes, there is a need for more elaborate hints, like those found in the *TrueType* language [2]. Such hints, which are tuned to the individual letter shapes, cannot be generated easily by a general purpose topological shape analyser. The model-based approach presented here enables advanced meta-hints to be associated with the model description of individual letter shapes. After having matched a given input font to the model, corresponding meta-hints are adapted to the features of the input font (sans-serif, italic) and reported into its outline description.



External outline :

ModelPt0	=	InputPt1
ModelPt1	=	InputPt1
ModelPt2	=	InputPt1
ModelPt3	=	InputPt2
ModelPt4	=	InputPt2
ModelPt5	=	InputPt2
ModelPt6	=	InputPt3
ModelPt7	=	InputPt5
ModelPt8	=	InputPt8
ModelPt9	=	InputPt8
ModelPt10	=	InputPt11
ModelPt11	=	InputPt0

Figure 2: Association of input shape support points with characteristic model points

This paper proposes a topological font-independent model for the description of each letter shape. This model incorporates a loose outline description valid for all non-fancy instances of the corresponding letter shape. Model outlines are described by characteristic points and by their interconnecting outline segments. Characteristic points are essentially either local maximas or minimas or discontinuity points. They specify starting, intermediate and ending points of different shape parts (stems, serifs, shoulders, bowls, diagonals). Their location within the character shape is given by their relative position in the plane. Outline segments are specified by their relative length, their approximate orientation and the way they connect to neighbouring segments. Relevant knowledge can be associated with characteristic points and their interconnections. For example, structure elements like half-serifs can be defined by a series of characteristic points (starting point, intermediate points, ending point). Meta-descriptions of grid constraints referencing characteristic points can easily be incorporated into the topological model.

This paper addresses the problem of identifying characteristic points in given character shapes. In order to solve it, we propose to match the support points of existing real letter shapes to their corresponding characteristic points in the model (figure 2).

Automatic labelling of character parts becomes a shape and point matching problem. A given input character is matched with the correspondent model character, its glyphs and contours are matched to the model character's glyphs and contours. By using the topological rules associated with each characteristic point, candidate points in the input description are successively checked, kept or eliminated. A successful match leads to one point in the input character for each characteristic model point.

For the purpose of automated hint generation, points of a given input font are matched to characteristic points of the model font. Then, it becomes easy to adapt the constraint descriptions from the model font to the given input font and to replace the characteristic glyph, contour and point numbers by the corresponding input font glyph, contour and point numbers.

2 Topological modelling of letter shapes

The shape of latin characters has evolved over the last centuries but, with the exception of script and ornamental types, the topology of most character shapes has remained identical. In order to establish a topological model of latin characters, we have tried to define meaningful points on their outline (characteristic points). These points are generally either local extremas, or junction points between different contours parts. The topological model incorporates the topological essence of the shapes of the members of an alphabet. It should be valid for all its non-fancy instances, e.g. for the font families which can be found in type collections such as Berthold Types [4].

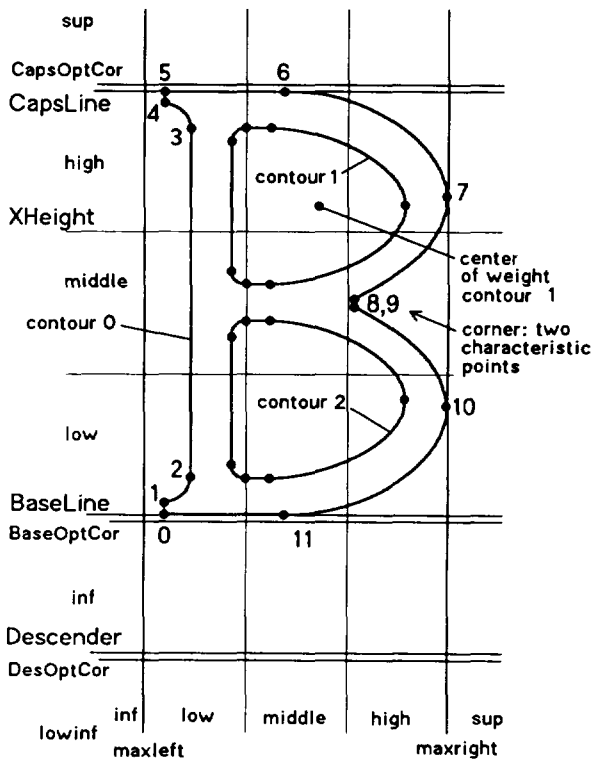
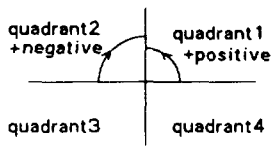
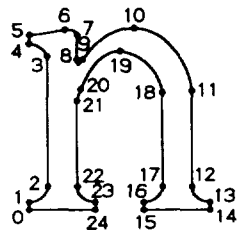


Figure 3: Loose representation of characteristic contour points of character "B"

Previous scientific approaches to topological modelling of letter shapes were established for the purpose of character recognition.





a) Synthetic model font

Pt0 : (low, base)
Pt1 : (low, inf+low)
Pt2 : (low, low)
Pt3 : (low, middle+high)
Pt4 : (low, high+sup)
Pt5 : (low, high+sup)
Pt6 : (low+middle, xheightoptcor)
Pt7 : (low+middle, high+sup)
Pt8 : (low+middle, high)
Pt9 : (low+middle, high)
Pt10 : (middle, xheightoptcor)
Pt11 : (high, middle+high)
Pt12 : (high, low)
Pt13 : (high, inf+low)
Pt14 : (high, base)
Pt15 : (middle+high, base line)
Pt16 : (middle+high, inf+low)
Pt17 : (middle+high, low)
Pt18 : (middle+high, middle+high)
Pt19 : (middle, high)
Pt20 : (low+middle, middle+high)
Pt21 : (low+middle, middle+high)
Pt22 : (low+middle, low)
Pt23 : (low+middle, inf+low)
Pt24 : (low+middle, base line)

b) Characteristic points

Curve0 : PT0 -> PT2 (footserif, up+right)
Curve1 : PT2 -> PT3 (long+vertical, up)
Curve2 : PT3 -> PT5 (headserif, up+left)
Curve3 : PT5 -> PT6 (diagonal, right)
Curve4 : PT6 -> PT7 (corner, down+right)
Curve5 : PT7 -> PT8 (vertical, down)
Curve6 : PT8 -> PT9 (corner, right)
Curve7 : PT9 -> PT10 (arc, quadrant2+negative)
Curve8 : PT10 -> PT11 (arc, quadrant1+negative)
Curve9 : PT11 -> PT12 (long+vertical, down)
Curve10 : PT12 -> PT14 (footserif, down+right)
Curve11 : PT14 -> PT15 (horizontal, left)
Curve12 : PT15 -> PT17 (footserif, up+right)
Curve13 : PT17 -> PT18 (long+vertical, up)
Curve14 : PT18 -> PT19 (arc, quadrant1+positive)
Curve15 : PT19 -> PT20 (arc, quadrant2+positive)
Curve16 : PT20 -> PT21 (corner, quadrant2+positive)
Curve17 : PT21 -> PT22 (long+vertical, down)
Curve18 : PT22 -> PT24 (footserif, down+right)
Curve19 : PT24 -> PT0 (horizontal, left)

c) Characteristic curve parts

Figure 4: Shape model of character "n"

They used topological features with a high discrimination potential and goodness evaluation parameters to match an input shape with unknown ASCII code to the correct character shape in the set of ASCII letter shapes [7] [8] [15]. Since our task consists in matching a given input letter shape with a known ASCII code to its reference model, the matching problem is limited to matching the shape's glyphs (individual shapes of a character), contours and characteristic points. A successful match however requires that each characteristic model point be matched without contradictions to one input shape outline support point.

In order to match glyphs, contours and characteristic points, the topological model incorporates rough positional information, as well as information concerning the position of a characteristic point relative to its neighbouring points. Unprecise positional information can be given by partitioning the character space into a very coarse coordinate grid. In the present implementation, 5 horizontal and 6 vertical subspaces (low-inferior, inferior, low, middle, high, superior) are used. This grid definition matches nicely the geometry of latin shapes: lower-case letters fit in a space defined by three horizontal and two vertical subspaces, capitals fit in a three by three space and special characters like the "$" use the full vertical space (6 subspaces). Characters extending beyond their left and right reference lines like italics use all 5 horizontal subspaces. Shape locations are defined in an unprecise way by specifying their approximate location on this coarse grid (figure 3). Further uncertainty can be introduced by allowing given model points to belong to one of two neighbouring coarse grid locations.

Glyphs and contour parts, like the interior contours of character "B" are situated on the grid by considering the position of their bounding-box center. Contour parts are also characterized as being exterior or interior contours, depending on their orientation.

Shape locations are specified in a precise way if they lie on certain reference lines like baseline, x-height, capsline and their respective optical correction lines. Characteristic points also have topological attributes in relation to the contour they belong to. They can be either local or global extrema in x or in y direction.

Pairs of neighbouring characteristic points are also characterized by the outline segment joining them. An outline segment is either a straight or nearly straight segment or a curve segment. Straight line segments are short or long. They have either a vertical or a horizontal primary direction. Curved segments have positive or negative orientation. They lie within a given quadrant (figure 4).

Relative positional information about glyphs and contours is used to match subparts of an input character shape to its corresponding model subparts (figure 5).

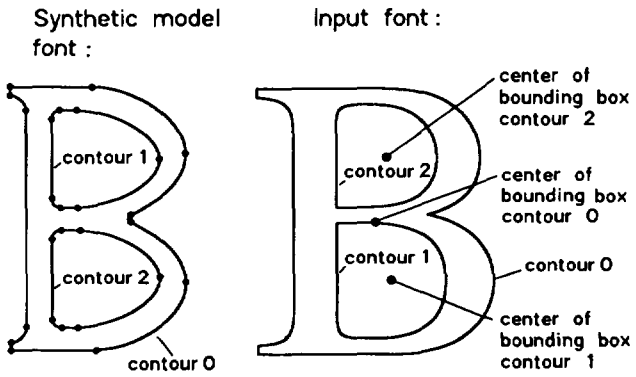
Glyphs and contours cannot be matched if topological contradictions are found between the input shape and the model shape. For example, character shape "%" cannot be matched with model character "%". Therefore the same character may have a model description including several shape variants. At matching time, either the variant is known in advance (figure 6) or the program tries to match the input shape successively to all variants, until a match without contradictions is found.

3 Matching characteristic points

As explained in the introduction, characteristic points define the different shape parts completely. In order to start the matching process, a list of matching candidate points of a given input font will be associated to each characteristic point of the model font. The selected candidate points must have the same approximate location as their corresponding characteristic point.

This first association based on the approximate geometric location of model points and shape points will produce many candidates for each characteristic model point.

Subsequently, a hierarchical succession of matching criteria is used in order to eliminate unsuitable candidates from the candidate list until one candidate remains as a match for one characteristic point of the topological description. The matching process proceeds successively from specific strong features to more global uncertain features of model letter shapes. If one or several candidates match a criterion associated with a characteristic point, they are kept in the candidate list and all other candidates who do not match this criterion are eliminated. If no candidate matches a given criterion, no candidates are eliminated and the matching process proceeds with the next criterion.



Model contour positions :
Contour 0 : (middle, middle)
Contour 1 : (middle, up)
Contour 2 : (middle, down)

Model font contour sequence :
hor. direction : (ModelCont0 = ModelCont1 = ModelCont2)
vert. direction : (ModelCont2 < ModelCont0 < ModelCont1)

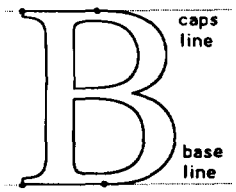
Input font contour sequence :
hor. direction : (InputCont0 < InputCont2 < InputCont1)
vert. direction : (InputCont1 < InputCont0 < InputCont2)

Contour matching list :
ModelCont0 = InputCont0
ModelCont1 = InputCont2
ModelCont2 = InputCont1

Figure 5: Matching input contours of character "B" to correspondent model contours

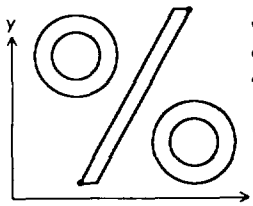
Candidate acceptance is based on the following topological criteria:

Criterion 1:



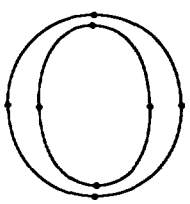
candidate point belongs to a reference line (base, x-height, capital lines)

Criterion 2:



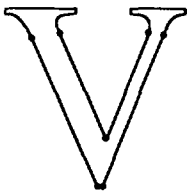
within a given contour, candidate point is a local extremum in x *and* y

Criterion 3:



within a given contour, candidate point is a global extremum in x *or* y

Criterion 4:

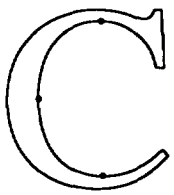


candidate point is a departure or an arrival point of a long straight line segment

Criterion 5:

candidate point is not situated between straight line segments having similar slopes

Criterion 6:

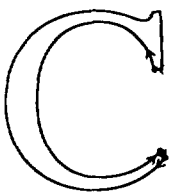


candidate point is a local extremum in the x or y direction

Criterion 7:

candidate point is both an arrival and a departure point of two oriented straight line segments with given directions

Criterion 8:



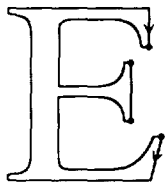
candidate point is both an arrival and a departure point of two oriented straight line or curve segments with given directions

Variant 1	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Variant 2	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Variant 3						t	w			W					&	

Times	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Helvetica	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Lucida	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Lucida Sans	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Courier	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Madeleine	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Avant Garde	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$
Haas Unica	a	g	i	j	s	t	w	J	S	W	3	6	9	%	&	$

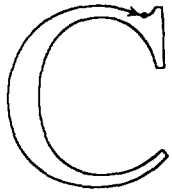
Figure 6: Examples of fonts having different shape variants

Criterion 9:



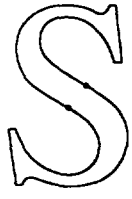
candidate point is an extremity point of an oriented straight line segment with given direction

Criterion 10:



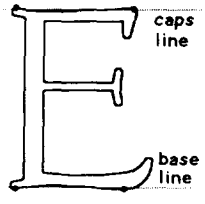
candidate point is an extremity point of an oriented straight line or curve segment with given direction

Criterion 11:

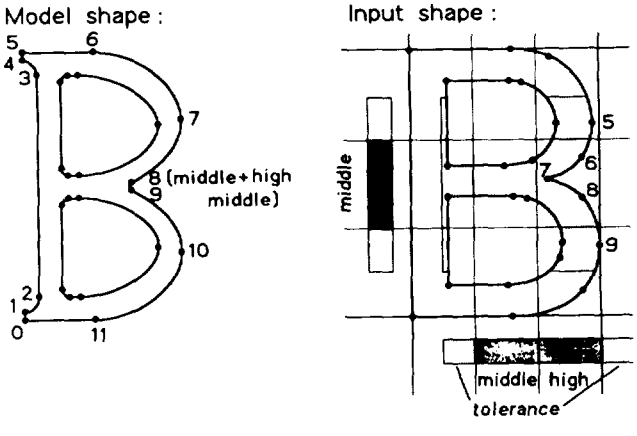


candidate point is an inflexion point

Criterion 12:

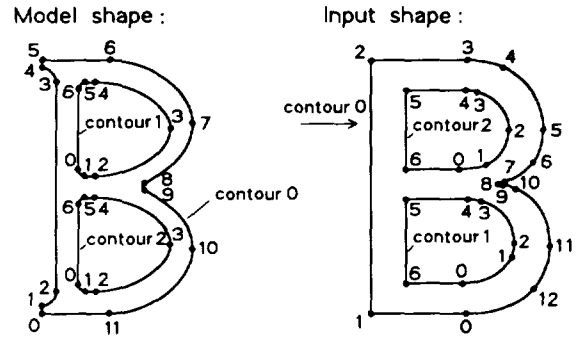


candidate point lies close to a *reference line* (baseline, descender line, x-height line, capital line)



Candidates for ModelPt8 on external contour:
Criterion: geometric location :
(InputPt5, InputPt6, InputPt7, InputPt8, InputPt9)
Next applicable criterion :
Local extrema in x- or in y-direction,
==> Selection of InputPt7 (Xmin)

Figure 7: Matching the intersection point of two arcs in character "B"



ModelCont0 = InputCont0	
ModelPt0	= InputPt1
ModelPt1	= InputPt1
ModelPt2	= InputPt1
ModelPt3	= InputPt2
ModelPt4	= InputPt2
ModelPt5	= InputPt2
ModelPt6	= InputPt3
ModelPt7	= InputPt5
ModelPt8	= InputPt8
ModelPt9	= InputPt8
ModelPt10	= InputPt11
ModelPt11	= InputPt0

ModelCont1 = InputCont2	
ModelPt0	= InputPt6
ModelPt1	= InputPt6
ModelPt2	= InputPt0
ModelPt3	= InputPt2
ModelPt4	= InputPt4
ModelPt5	= InputPt5
ModelPt6	= InputPt5

ModelCont2 = InputCont1	
ModelPt0	= InputPt6
ModelPt1	= InputPt6
ModelPt2	= InputPt0
ModelPt3	= InputPt2
ModelPt4	= InputPt4
ModelPt5	= InputPt5
ModelPt6	= InputPt5

Figure 8: Characteristic point table includes the correspondences between input shape points and model

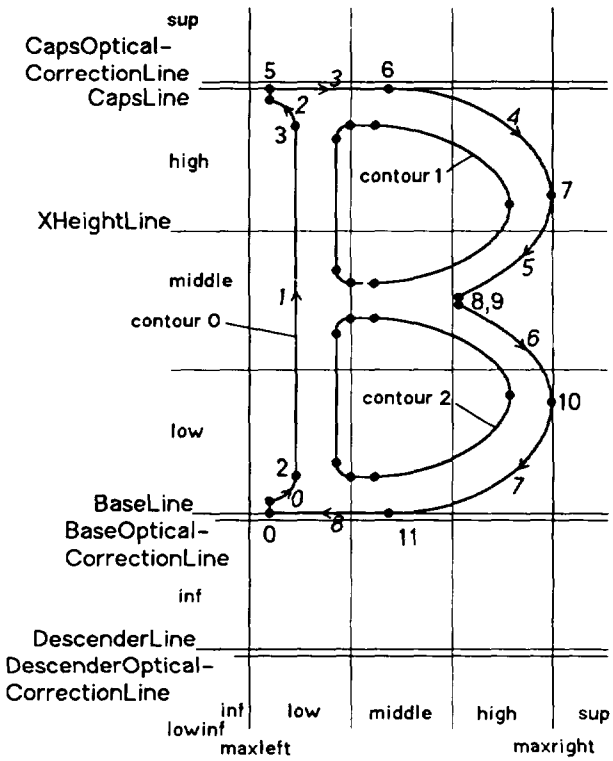
The model description associates characteristic points with information about their approximate location and about their topological properties (local or global extrema, position on or close to a given reference line, direction of arrival and departure segments). The model also includes outline part descriptors (figure 9). These descriptors specify departure and arrival point numbers of given outline segments and their type (straight, curved of serif). Straight line segments are characterized by their primary direction (horizontal, vertical or diagonal) and orientation. Curve segments are characterized by their respective quadrant number and orientation. Outline segments can be short or long segments.

The previously enumerated criteria for the matching of characteristic points are hierarchical. Stronger and more specific features are tested first (location on reference lines, local extrema in x and y, global extrema in x or y). If applicable to a given model point, they result in rapid elimination of inappropriate candidate points.

Criteria (4) to (5) incorporate knowledge about incoming and outgoing outline segments. Again more specific criteria are tested first (criterion 4: extremity of a long straight line segment). Criterion (5) is used in order to remove candidate points, since model points do not lie between straight line segments having similar slopes. Criteria (7) to (10) use the directions of incoming and outgoing outline segments for the further discrimination of input points. Criteria (11) and (12) are two remaining criteria which provide additional information in order to find the best match between the few remaining candidate points.

After applying each criterion and eliminating unsuitable candidate points, the outline description of the remaining candidates is checked for coherence. Contradictions are detected and inappropriate candidate points are removed from the candidate point list.

Figures 7 and 10 illustrate the matching process for associating input points to corresponding model points.



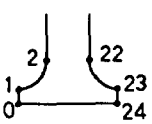
Features associated with points :

Point 0:	(low,baseline),	local minimum in y
Point 1:	(low,inf+low),	local minimum in x
Point 2:	(low,low),	local maximum in x
Point 3:	(low,high),	local maximum in x
Point 4:	(low,high+sup),	local minimum in x
Point 5:	(low,capsline),	local maximum in y
Point 6:	(middle,capsline),	local maximum in y
Point 7:	(maxright,high),	local maximum in x
Point 8,9:	(middle+high,high),	local minimum in x
Point 10:	(maxright,low),	global maximum in x
Point 11:	(middle,baseline),	local minimum in y

Features associated with outline parts :

contour part 0:	serif,	right+up
contour part 1:	long+vertical,	up
contour part 2:	serif,	up+left
contour part 3:	long+horizontal,	right
contour part 4:	arc,	quadrant1+negative
contour part 5:	arc,	quadrant4+negative
contour part 6:	arc,	quadrant1+negative
contour part 7:	arc,	quadrant4+negative
contour part 8:	long+horizontal,	left

Figure 9: Model description of character B

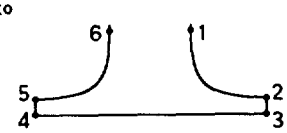


Model font

ModelPt0 :
- point located on the base line
- local minimum in y-direction
- arrival of a horizontal right to left segment
- point located close to base line

ModelPt1 :
- local minimum in x-direction
- intermediate point within a serif

ModelPt2 :
- departure of a long vertical low to high segment

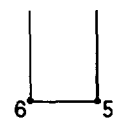


Input font:
Times Roman

ModelPt0 = InputPt4
- point located on the base line
- local minimum in y-direction
- arrival of a horizontal right to left segment

ModelPt1 = InputPt5
- local minimum in x-direction
- intermediate point within a serif

ModelPt2 = InputPt6
- departure of a long vertical low to high segment



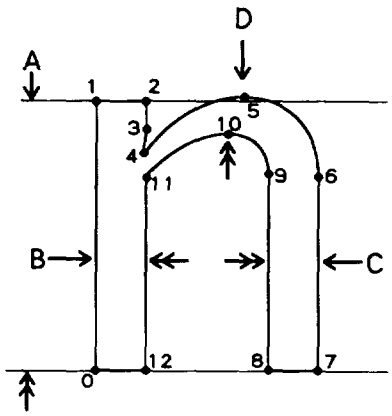
Input font:
Helvetica

ModelPt0 = InputPt6
- point located on the base line
- local minimum in y-direction
- arrival of a horizontal right to left segment

ModelPt1 = InputPt6
- intermediate point within a serif

ModelPt2 = InputPt6
- departure of a long vertical low to high segment

Figure 10: Matching serifs extremities



A: **hint specification:** vertical phase control of reference lines
hint application: complete character

B: **hint specification:** horizontal phase control of vertical stem
stem width given by Pt0, Pt12
hint application: complete character

C: **hint specification:** horizontal phase control of vertical stem
stem width given by Pt8, Pt7
hint application:
fixed displacement: Pt6 to Pt9
proportional displacement: Pt4 to Pt6
fixpoint: Pt4
max. displacement point: Pt6
proportional displacement: Pt9 to Pt11
fixpoint: Pt11
max. displacement point: Pt9

D: **hint specification:** vertical phase control of shoulder with
respect to reference lines
shoulder thickness given by Pt10, Pt5
hint application:
proportional displacement: Pt4 to Pt6
fixpoint: Pt4, Pt6
max. displacement point: Pt5
proportional displacement: Pt9 to Pt11
fixpoint: Pt9, Pt11
max. displacement point: Pt10

Figure 11: Support points for the specification of grid constraints

The result of the matching process is fed back into the input shape outline description as a characteristic point table (figure 8). Each table entry specifies a given model point and its associated input point given by its glyph, contour and point number.

A model description has been established for all latin alphanumeric characters. Italic characters do not need a special model description: before matching, they are rectified by inverse slanting. After rectification, the normal matching process can be applied to them. The matching program has been tested on 70 different fonts, including italic fonts. Among these fonts, a full match has been obtained on 99% of all characters. For the remaining 1% of the characters, the program announces that no correct match has been found. Characters which could not be matched to the model belong to fonts having slightly rounded vertical, horizontal and diagonal strokes like Optima, Palatino or Zapf Book. Such characters require the implementation of additional processing steps for assimilating their long low-curvature outline parts to long vertical, horizontal or diagonal outline segments.

4 Automatic hinting

Hints are grid-fitting rules specifying which parts of outline characters should be adapted to the grid in order to obtain nice regular raster characters [11]. These grid-fitting rules mainly apply to character reference lines, stems, bowls and serifs. Grid-fitting rules for fitting stems and bowls require two outline support points specifying the stem or bowl width.

The evaluation of a given hint produces a subpixel displacement which must be applied to certain parts of the outline. The application part specifies the character parts on which the computed displacement is applied by giving their starting and ending outline support points (figure 11).

Similarly, special grid-fitting rules control the discrete appearance of serifs. They require outline support points for the control of serif length and serif thickness (figure 12).

Thanks to grid-fitting, the discrete size of serifs will decrease continuously and disappear at once when decreasing font size (figure 13).

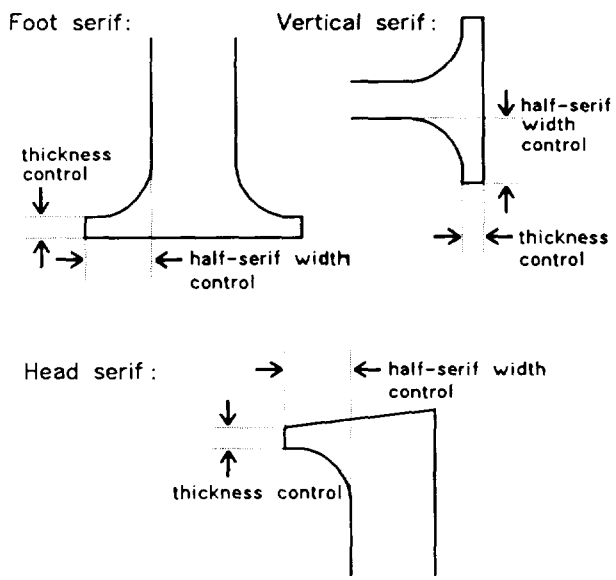


Figure 12: Serif control hints

Only those hints whose parameters are characteristic points of a given letter shape can be generated automatically. Their meta-description is inserted into the character model. When the correspondence between characteristic points of the model and points of the input shape has been found, applicable hints described in the model are copied into the input shape description and model point numbers are replaced by references to corresponding input shape point numbers.

Some input shapes only partially match model shapes. Matching sans-serif fonts with the model produces one input point for several characteristic points at serif locations (figure 10). Degenerated serifs are detected by the automatic hinting procedure. Hints whose parameters are two identical points are not copied into the input

shape description.

As mentioned in the previous section, italic characters are matched to their model after applying inverse slanting. Meta-hints associated with the model are defined in a way which ensures that correct hints can be derived both for upright and italic typefaces. For the vertical phase control of horizontal bars, hint specifications remain essentially the same: the current displacement direction will follow the direction of the vertical stems (figure 14). Support points used for horizontal phase control of vertical or italized stems can be defined in such a way that hints for both cases derive from the same meta-hint description.

Italic serif shapes differ considerably from roman serifs. Such shape variations can be detected and meta-hints associated to such shapes need not be copied into the input typeface description (figure 15).

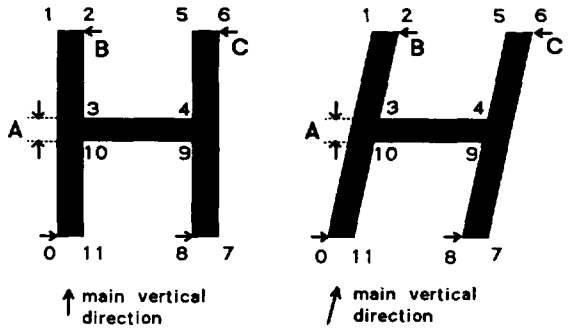
Figure 13: Serif appearence with decreasing font size

Using the same topological description and identical meta-hints for upright and italic typefaces, one achieves quite acceptable results (figure 16).

Characters including features not described by the model description are hinted manually. These hints are very special. Generally, they can be left out since their effect is rather limited. Such hints may be used for example for controlling the tail of character "Q" or the terminal drop of character "j". The number of special hints that may be added explicitely is smaller than 3%. The automatically generated hints already produce high quality character descriptions that can be rendered at any given screen or printer resolution.

5 Character structure elements

Most computer-aided font design systems incorporate either interactive or descriptive outline manipulation capabilities. Shape parts like shoulders, stems and serifs need to be specified and stored explicitely. These parts may then be reused for the synthesis of other similar letter shapes [14]. *Metafont* for example is a typographic synthesizing system based on a programming model that is used to superimpose parametrised character parts in order to generate the resulting letter shape. Parameters for defining and assembling character parts are specified explicitely using suitable expressions [13]. The complete shape is generated as a superimposition of each individual shape. Metafont however provides no tools for extracting character parts from existing outline fonts. This section shows how our topological model is used in order to build higher level structure elements from outline descriptions for describing character parts like serifs and stems.



A: hint specification:
vertical phase control of horizontal bar
bar width given by Pt10, Pt3
hint application:
vertical displacement along main direction of horizontal bar: $\overline{Pt3Pt4}$, $\overline{Pt10Pt9}$

B: hint specification:
horizontal phase control of vertical stem;
stem support points given by Pt0, Pt2
hint application:
displacement of stem borders $\overline{Pt0Pt1}$, $\overline{Pt3Pt2}$, $\overline{Pt10Pt11}$
if vertical stem: horizontal phase control only
if oblique stem: horizontal and diagonal phase control

Figure 14: Common hints for upright and italic characters

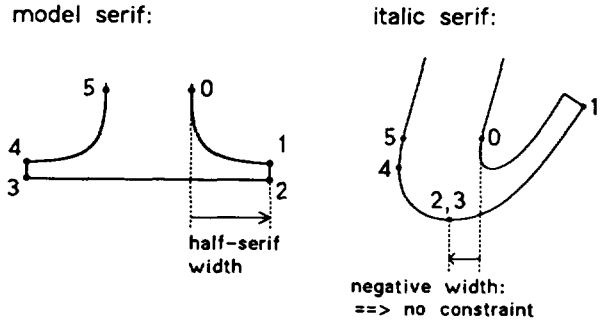
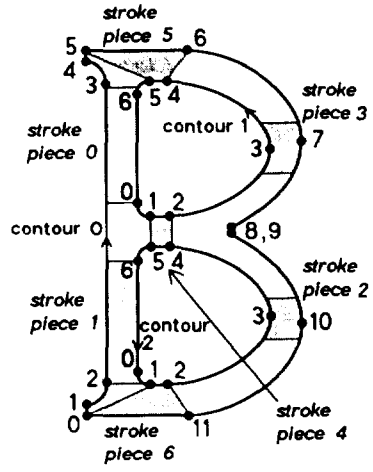


Figure 15: Detection of important shape variations in italic serifs

Serifs are specified directly by the topological model as outline parts having the *serif* attribute (see figure 4). In fact, each head serif is given by one and each foot or vertical serif is given by two half-serifs (see figure 12). Each half-serif is given by a starting point lying at the end of a long vertical or horizontal bar, by one intermediate point giving the serif's extension and by an end point lying on the continuation of the stroke base. A complete foot serif is given by its two component half-serifs. With such a description, a program can automatically extract the serifs of a given outline font. For uniformization purposes, new regularized serifs can be inserted in place of the original ones. Furthermore, vertical and horizontal stems and bowl pieces embedded in the character outline can be described by specifying corresponding stroke pieces (figure 17).

Description of stroke pieces may be useful for adjusting stroke thick-




```
Stroke piece 0:
  Vertical stem piece
    contour 0: segment (Pt2 to Pt3)
    contour 1: segment (Pt6 to Pt0)
    continuity: with stroke piece 1
Stroke piece 1:
  Vertical stem piece
    contour 0: segment (Pt2 to Pt3)
    contour 2: segment (Pt6 to Pt0)
    continuity: nil
Stroke piece 2:
  Vertical bowl part
    contour 2: arc extremity: Pt3
    contour 0: arc extremity: Pt10
    continuity: nil
Stroke piece 3:
  Vertical bowl part
    contour 1: arc extremity: Pt3
    contour 0: arc extremity: Pt7
    continuity: nil
Stroke piece 4:
  Horizontal stroke piece
    contour 2: segment (Pt4 to Pt5)
    contour 1: segment (Pt1 to Pt2)
    continuity: nil
Stroke piece 5:
  Horizontal stroke piece on CapsLine
    contour 1: segment (Pt4 to Pt5)
    contour 0: segment (Pt5 to Pt6)
    continuity: nil
Stroke piece 6:
  Horizontal stroke piece on BaseLine
    contour 0: segment (Pt11 to Pt0)
    contour 2: segment (Pt1 to Pt2)
    continuity: nil
```

Figure 17: Description of stroke pieces embedded in the model character outline

Figure 16: Rasterization of automatically hinted italic outline characters

ness and position, which is important for producing more legible fonts at small sizes.

6 Conclusions

A topological model representing the essence of the shapes of typographic latin typefaces has been developed. This model provides sufficiently general information for it to be valid for all non-fancy typefaces, serif and sans-serif. It also provides sufficient topological information and relationships to match typefaces given by their outline description to the model shape.

The correspondence between characteristic model and input shape points is of great importance for further processing of character outline descriptions. Grid-fitting meta-hints can be taken automatically from the model, adapted and associated to any given input typeface. Furthermore, higher-order structures can be built upon characteristic points for describing structural character parts like serifs, stems, bowls and junctions. Mapping continuous outline descriptions into structural descriptions and vice-versa offers new opportunities for developing advanced computer-aided font design tools.

Acknowledgements

The authors would like to thank Andre Gurtler from the School of Design, Basel, for his contribution to visual aspects of digital type. We would also like to thank Jakob Gonczarowski and Justin Bur for their critical review of our typographic font-independent topological model. This research was funded by the "Commission d'Encouragement de la Recherche Scientifique" of Switzerland.

Bibliography

[1] D. Adams, "abcdefg: a better constraint driven environment for font generation", in Andre, Hersch (eds.), *Raster Imaging and Digital Typography*, Cambridge University Press, 1989, 54-70

[2] Apple Computer, *TrueType Spec – The TrueType Font Format Specification*, July 1990

[3] S. Andler, "Automatic Generation of Gridfitting Hints for Rasterization of Outline Fonts or Graphics", *EP90 – Proceedings of the International Conference on Electronic Publishing, Document Manipulation & Typography*, September 90, (R. Furuta, Ed.) Cambridge University Press, 221-234

[4] *Berthold Types*, H. Berthold AG, Berlin, 1988

[5] C. Betrisey, R.D. Hersch, "Flexible Application of Outline Grid Constraints", in Andre, Hersch (eds.), *Raster Imaging and Digital Typography*, Cambridge University Press, 1989, 242-250

[6] P. Coueignoux, "Character Generation by Computer", *Computer Graphics and Image Processing*, Vol. 16, 1981, pp 240-269.

[7] M. Eden, "Handwriting and pattern recognition", *IRE Trans. Inform. Theory*, Vol IT-8, 1962, pp 160-166.

[8] H.F. Feng, T. Pavlidis, "Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition", *IEEE Transactions on Computers*, Vol C-24, June 1975, pp 636-650.

[9] J. Flowers, "Digital type manufacture: an interactive approach", *IEEE Computer*, May 1984, pp. 40-48.

[10] P. Gaskell, "A Nomenclature for the Letterforms of Roman Type", *Visible Language*, Vol 10, No 1, 1976, 41-51.

[11] R.D. Hersch, "Character Generation under Grid Constraints", Proceedings SIGGRAPH'87, *ACM Computer Graphics*, Vol 21, No. 4, July 1987, 243-252

[12] P. Karow, *Digital Formats for Typefaces*, URW Verlag, Hamburg, 1987.

[13] D. Knuth, *Computer Modern Typefaces*, Addison-Wesley, 1986.

[14] R. Rubinstein, *Digital Typography, An Introduction to Type and Composition for Computer System Design*, Addison-Wesley, 1988.

[15] L.G. Shapiro, "A Structural Model of Shape", IEEE PAMI, Vol PAMI-2, No 2, March 1980, pp 111-126.

[16] W. Tracy, "Letters of Credit, a view of type design", Gordon Fraser, London, 1986, pp 52-55.

Annex: Matching typographic letter shapes to their topological model