A Hybrid Approach to Rendering Handwritten Characters

Sara L. Su
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA, 02139, USA
sarasu@mit.edu

Chenyu Wu
Carnegie Mellon University
Robotics Institute
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
chenyuwu@cmu.edu

Ying-Qing Xu,
Heung-Yeung Shum
Microsoft Research Asia
5/F, Beijing Sigma Center
No.49 Zhichun Road, Hai Dian
Beijing, China 100080
{yqxu,hshum}@microsoft.com

ABSTRACT

With the growing popularity of pen-based computers comes the need to display clear handwritten characters at small sizes on low-resolution displays. This paper describes a method for automatically constructing hinted TrueType fonts from on-line handwriting data. Hints add extra information to glyph outlines in the form of imperative constraints overriding side effects of the rasterization process. We use an aggressive matching strategy to find correspondences between an input glyph and a previously-hinted template, considering both global and local features to allow hinting even when they differ in shape and topology. Recognizing that stroke width statistics are among features that characterize a person's handwriting, we recalculate global values in the control value table (CVT) before transfer to preserve the characteristics of the original handwriting.

Keywords

Handwriting, automatic hinting, digital typography, shape matching, pen-based interaction.

1. INTRODUCTION

Handwriting plays an integral role in our thought processes, functional tasks, and communication with peers, and perhaps even offers some insight into personality traits [Bra91]. How we write, along with what we write, defines who we are.

With all that we rely on handwriting for, it is perhaps unsurprising that pen-based computers are growing in popularity. Appearing as small handheld devices, personal tablet computers, and large whiteboard displays, numerous systems since Sketchpad [Sut63] have demonstrated stylus-based interaction to be a concise, effective means of user input.

While many handhelds accept character-by-character input as stylized "graffiti" [Mac97], as the popularity of pen-based computing continues to grow, an increasing number of people will rely on applications with freehand input. Advertisements for tablet computers, targeting users who work away from the desk, tout them as being as natural to write on as a pad of paper.

Much work has been done in the areas of recognition [Mac94], simulation [Dev95], and learning-based synthesis of handwriting [Guy96, Wan02], but less attention has been paid to the problem of rendering the resulting characters on screen. Whether they were synthesized, scanned, or written directly onto a tablet screen, digital handwriting must at some point be rendered legibly and without loss of quality.

Recognizing the demand for onscreen text that is both readable and unique to the user, digital type foundries have begun offering "personal handwriting fonts", typefaces designed based on a customer's signature or writing samples. Like other typefaces, some of these fonts contain essential gridfitting instructions, *hints*, that specify the appearance of characters at varying point sizes and display resolutions. While some handwriting fonts are manually hinted (an extremely time-consuming task), most are either hinted automatically by a typeface authoring system such as Macromedia Inc.'s Fontographer or contain no hints at all. While Fontographer's auto-hinting system is effective for traditional typefaces of size 24 pt or larger, handwritten glyphs are a special case that most existing auto-hinters do not handle well.

Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972
WSCG 2004, February 2-6, 2003, Plzen, Czech Republic.

We propose a hybrid method for automatically hinting handwriting by considering global and local features of each glyph against knowledge of already-hinted templates. Though in this paper we discuss these techniques in the terms of TrueType template fonts and hint instructions, we see them as applicable in the more general context of intelligent rendering of handwriting. Here we implement the specific case of converting handwritten characters from a polylines on a tablet device to TrueType glyphs. The results are encouraging and suggest an application scenario in which the user can create a more legible form of his or her own handwriting directly on a tablet without having to wait for a company to digitize and hint writing samples as a typeface.

2. BACKGROUND

Although many alternative representations have been proposed [Knu86, Kla93, McG95], outline fonts are still the format most widely used today. Outlines avoid many of the problems that plagued earlier bitmap fonts (every required size must be hand designed, they are tuned to a specific printer, and the footprint of a font grows quickly with the size of the characters), but to be displayed on screen, they must eventually be converted to bitmaps [Rub88].

Hinting gives a typographic engineer fine control over the appearance of glyphs when rasterized. With these gridfitting instructions, the typographer specifies constraints between knots of a glyph or between a knot and a gridline. Though it is a laborious task, hinting is essential for legible rendering of glyphs. Stroke width uniformity, stroke continuity, glyph spacing: all are controlled by hinting. The difference in quality between hinted and unhinted glyphs is most apparent for small point sizes displayed at typical screen resolutions of 72-120 dpi. Hinting also improves the appearance of small text faxed at 200 dpi or printed at 300-600 dpi [Sta97].

The two major font standards, TrueType and Postscript (or Type 1), though both using outline representations of characters, incorporate two very different hinting philosophies. While Postscript fonts leave control of a character's final appearance to the rasterizer [Ado90], a typographer embeds explicit gridfitting instructions in the outline description when designing a TrueType font [App96, Con97, Typ96].

2.1 Postscript hinting

In the description of a Postscript font, semantic features of each glyph are marked, and hints contain information about vertical and horizontal bands across these features. It is up to the rasterizer to use this information to optimize the distribution of pixels by stretching or compressing glyph outlines within the defined bands. Because control of the character's final appearance falls to the rasterizer, the typographer cannot specify exactly what it will look like when rendered. However, the relative simplicity of Postscript hints makes it more straightforward to develop automatic hinting systems based on recognition of semantic features.

2.2 TrueType hinting

In TrueType, there is no concept of bowls, stems, or other semantic features of a character; there are only knots and splines. The designer of a TrueType font can control the precise layout of a glyph's pixels at a particular size by programming explicit gridfitting instructions into the description of the font. Tools such as Fontographer and Visual True-Type [Sta98] generate hint instructions in high-level, declarative languages that are then compiled to the TrueType assembly language. Like Zongker et al. [Zon00], we discuss hint translation in terms of the VTT Talk language provided by Visual TrueType [Mic97].

A single VTT Talk instruction specifies a constraint between two knots in a glyph, between a knot and a gridline, or on a group of knots in a contour. The following types of VTT Talk hints are defined: An *anchor* rounds a parent-less knot to the grid or to a gridline specified by a CVT entry. A child knot's position is maintained relative to its anchored parent with the use of *distance* and *link* constraints. A distance constraint specifies the absolute distance to maintain while a link refers to a CVT entry. Both parent and child are rounded to gridlines such that there is a minimum distance of 1 pixel between the two. A child knot's position is maintained relative to two parents with an *interpolate* instruction. A *shift* maintains a child's distance to its parent even if hinting has moved the parent. Unlike with a link, the child's position is not rounded to the grid, thereby allowing movements of less than a full pixel. *Deltas* and *moves*, known as exceptions, are used to specify the exact number of pixels at a point at a particular glyph size. A delta affects a single size while a move applies to all sizes of a glyph.

There has been significant earlier work on automatic "tuning" of typefaces including [Her91, Hob93, Her94, Zon00, Sha03].

Hersch and Bétrisey [Her91] developed model-based methods for automatic hinting, transferring gridfitting instructions from specially constructed intermediate models. The model for each glyph includes both an outline description of shape as well as a listing of its semantic parts. After matching the outlines of the glyph to be hinted to those of the

model, the semantic features of the target glyph can be labeled and hints generated.

Zongker et al. [Zon00] adapted this work to create a production tool for hinting TrueType fonts. Rather than using a manually constructed model as a bridge between knots on the outline character and the semantic features needed for hinting, their method uses an already-hinted TrueType font as the template. The template can be cleverly chosen to be a good match to the target font, resulting in good quality hints. The instructions transferred using this method retained the hinting techniques particular to the individual typographer.

3. METHOD

Our hinting method is motivated by earlier work on model-based shape matching [Her91] and example-based hinting of TrueType fonts [Zon00]. These automated hinting systems transferred instructions from a manually-hinted template to a new input glyph. We build on techniques introduced in these systems to automatically hint handwritten glyphs that often differ from the predefined templates.

The first step is to determine correspondences between template and input knots. We first calculate global correspondences between a glyph and the same glyph from the template set and then calculating local correspondences through comparisons to analogous curves of other template glyphs. This hybrid approach allows us to find matches even for input/template glyph pairs that are topologically very different.

After knot correspondences have been found, hint instructions are translated from template to input in a relatively straightforward process. In addition to glyph-specific hints, global data in the *control value table* (CVT) used to unify structural elements across glyphs are also translated. One could argue that the CVT is not useful when dealing with the irregularities of handwriting. However, though the constraints are hardly as rigid as those of traditional typefaces, there still exists a degree of uniformity across characters in most handwriting. Indeed, it is these patterns and shared features that aid a reader in identifying the familiar handwriting of a friend. We recompute values in the CVT based on measurements at input glyph knots, creating new CVT entries for features not sufficiently captured in the template.

3.1 From strokes to points and curves

Many fonts originating from brushed or penned strokes take their glyph shapes from the physical acts of creating them. Unlike many traditional typefaces, the appearance of script, calligraphic, or "handwriting-like" glyphs has more to do with letter formation patterns than with intentional typographic form.

Our goal is to preserve the characteristic stroke widths of handwritten characters using hints. In this section, we describe the process of extracting outline knots of a variable-width handwritten stroke.

3.1.1 Reconstructing variable-width strokes

Input strokes could come from a variety of sources: scanned from paper, created in a digital painting program, or input directly from a tablet device. Currently, most pen-input devices render handwriting as fixed-width polylines. However, most do record physical information, such as direction and speed of pen movement, that can be used to reconstruct the variable-width stroke as it might appear on paper.

To simulate pen movement, we use a straightforward physical model for rendering the pen strokes with variable width. We assume that the pen's movement involves only translation and regard the pen tip as a perfect circle at the point of initial contact. As the pen moves, extrusion forces in the x- and y-directions cause the circle to deform into an ellipse of constant area as illustrated in Figure 1.

In addition to the direction and speed of pen movement, pen pressure is taken into account in calculating the deformation of the virtual pen tip. Rendering the changing position and shape of the ellipse through time produces the variable-width stroke from whose outline knots can be extracted to create a TrueType glyph.

This deformation method works well for reconstructing a variable-width, brush-like stroke. Arabic language fonts, as well as some Latin calligraphic fonts, require a different model. In these cases, the pen tip is rigid, and it is the nib angle and direction of pen movement that determine the stroke thickness.

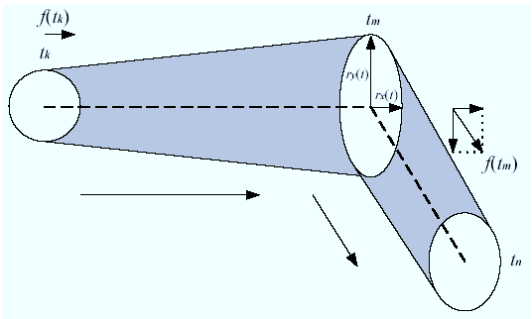

Figure 1. Extrusion forces deform the pen tip into an ellipse as the pen moves. $f(t_m)$ indicates the extrusion force at t_m.

3.1.2 Curve-fitting

By sub-sampling the outline of a variable-width input stroke, we can extract all control points to form the point-and-curve description of the TrueType glyph. TrueType outlines are defined by on-curve and off-curve points. Adjacent on-curve points are connected with straight line segments while off-curve points, along with neighboring on-curve points, define Bézier curve segments. In this case, we only use on-curve points. Though their use results in a larger-footprint outline description, this larger set of on-curve points preserves more of the topology of the input character. In the future, we may pursue alternative curve-fitting techniques capable of also approximating off-curve points, resulting in a smaller-footprint outline description. The points are renumbered according to their location on glyph contours before being written to the final font file. These points will be used in the outline glyph definition in the TrueType font file to be manipulated by the auto-hinting processes.

3.2 Correspondence search

In order to transfer hints from a template character set (an already-hinted font), we must determine the correspondence between the template font and input glyphs, or more specifically, between the template and input knots.

We first attempt to match the overall topology of an input glyph with the corresponding template glyph.

```
proc FindCorrespondences(Glyph G_i, Glyph G_t )
    while ( |G_i| > |G_t| and G_i.hasCollinearStrokes( ) ) {
        //join the most collinear strokes of G_i
    }
    while ( |G_i| > |G_t| and G_i.hasCorners( ) ) {
        //split G_i at the sharpest corner
    }
    CorrespondenceSet C_min
    C_min.numKnots ← G_i.numKnots
    for each knot i in G_i {
        C_min.knots[0][i] ← G_i.knots[i]
    }
    C_min.energy ← ∞
    //consider all permutations of correspondences
    for each CorrespondenceSet C {
        C.energy ← 0
        for each (knot J, knot K) in C {
            C.energy ← C.energy
                    +(J.x-K.x)^2 +(J.y-K.y)^2
        }
        if ( C.energy < C_min.energy ) {
            C_min ← C
        }
    return C_min
```

Figure 2. Global search algorithm.

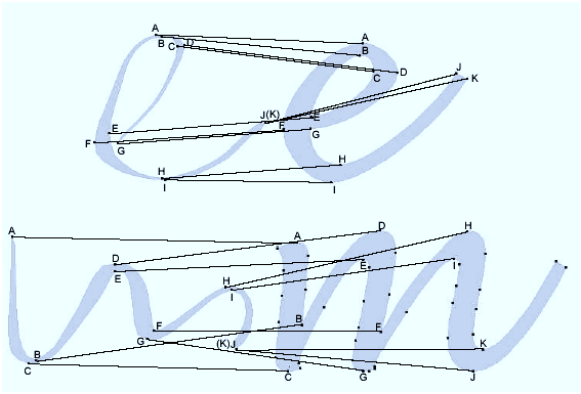

Figure 3. The global correspondence search attempts to match each knot on an input glyph with one on the corresponding template.

For reasonably similar template and input character sets, this global correspondence search is sufficient. However, for glyphs whose shapes differ significantly from their templates, more than a global topological search is required. In this case, we also perform a local search for correspondences in similar curves of different template glyphs.

3.2.1 Global search

Suppose we wish to hint an input glyph G_i based on its corresponding template glyph G_t. For each knot of G_t, we attempt to find an analogous knot in G_i.

We first attempt to balance the number of strokes with a strategy similar to that employed by [Arv00]. We join strokes of G_i that are nearly collinear and split those containing sharp corners. Note that this step does not physically split or join strokes; rather the strokes are merely hinted as though these operations have been applied. The rendered appearance of the character is not altered.

In order to maximize the number of hints transferred, we find a matching input knot for each on-curve template knot. If later a match is deemed inappropriate, the related hints can be ignored in the translation step. We consider all permutations of correspondences between knots. While earlier attempts to find the best correspondence have been primarily heuristic-based, our algorithm calculates the optimal correspondence based on the "energy" required for morphing the input character to the template, calculated as the sum of the squared distances between template and input knots. Though simple, this measure of cost is quite effective. In the future, it would be worthwhile to consider including other factors in the cost such as the energy required to distort glyph features during morphing. Alternatively, we could apply a physically-based shape-blending such as that described in [Sed92].

Information about the approximate location of each knot is used reduce running time. As a pre-

processing step, each glyph is segmented into four geographic regions, each knot being tagged with this information. Local energy is only calculated for pairs of template and input knots located in the same region.

With fairly uniform handwriting, a single template font is usually sufficient. However, as mentioned above, handwriting exhibiting a high degree of variance across glyphs cannot be accurately matched with a single template. Given a number of possible templates, we must choose the one most closely matching the input. Comparing each possible template against our input, we determine the best match to be the one with the least total energy.

3.2.2 Local search

Figure 3 shows the results of the global correspondence search for two pairs of glyphs. A complete set of correspondences can be found for the 'e' glyphs, with each template knot paired with an input knot. The match for the 'm' glyphs is less successful. A successful global correspondence search requires a high degree of similarity between two glyphs. When this is not the case, the global search will fail to find a complete match. In addition, a number of letters appear in multiple topological forms, for example lowercase 'a', 'g', and 'r', and uppercase 'I' and 'Q'. Such cases motivate the need for a local correspondence search that considers matches with other glyphs of the character set.

As a pre-processing step, template and input glyphs are split into component strokes based on the degree of curvature at each on-curve point. To approximate letter formation patterns, we determine stroke splits at knots with a high degree of curvature.

Each template we initially consider contains a component stroke that could possibly fit a section in the input glyph well. By analyzing the number of contours, start and end points, variation in the skeleton direction, and glyph region, we determine the template most closely matching the input.

Next, we calculate the feature points of the given contour in a three-step process. Using curvature to determine feature points results in many redundant points due to the large number of on-curve points in the input. Therefore, we consider only the most prominent feature points (maxima and minima) and map each of these to feature points in the input. Next, we map the pairs of feature points (manually labeled in the template) that we have found in the first step, with pairs extracted from the input. Finally, we map the remaining feature points in the template with the translated points in the input. Note that these translated points are selected from several candidate points by preserving most of the topological structure among feature points in the template. In this way, we maintain the original hinting style and accuracy.

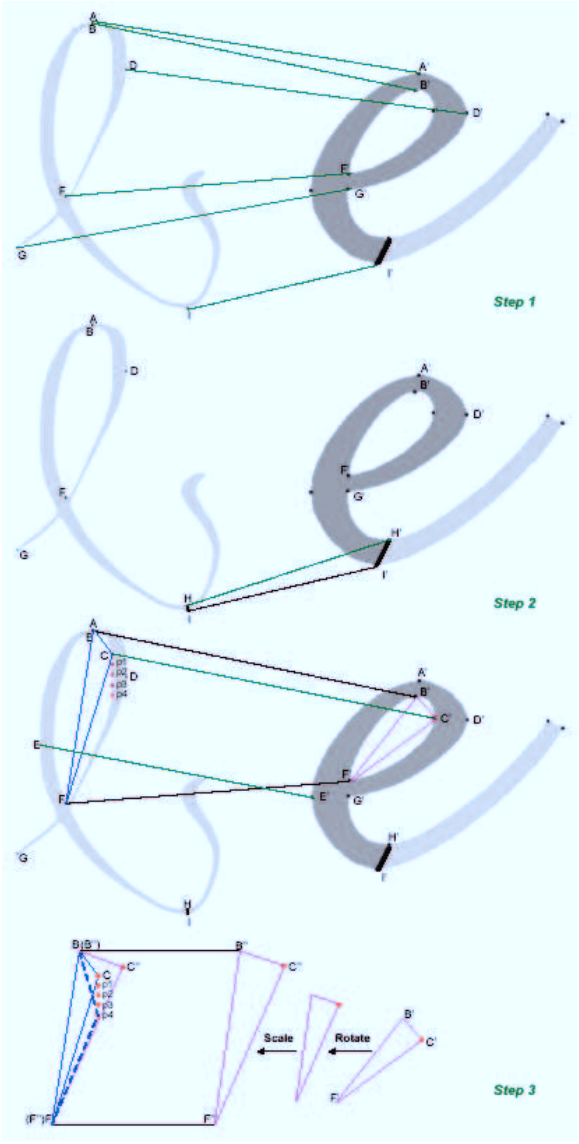

Figure 4. Steps in finding a local correspondence. (1) Feature points are identified in the x- and y-directions. (2) The analogous point to the feature point of interest is identified. (3) After matching B′ with B and F′ with F, we get the triangle B′C′F′. The sets of points B, C, F, and B′, C′, F′ define a unique affine transformation leading to a new triangle B″C″F″ with side B″F″ overlapping BF. By selecting a feature point from C, p1, p2, p3 and p4, with minimal distance from C″, a translated triangle BCF can be found that most closely matches the original triangle B′C′F′.

This algorithm is perhaps most easily discussed in the context of an example. Figure 4 illustrates the steps to finding the local correspondence between a template and input glyph.

3.3 Hint translation

After correspondences between input and template knots have been found, hint translation is relatively straightforward. Hint programs are copied from the templates and attached to the input glyphs, substituting corresponding knot numbers in the VTT Talk instructions. Hints involving a knot for which only a weak final correspondence was found are discarded.

We translate hint instructions that preserve location, distance, and proportions: Distances, links, and shifts maintain the width of a stroke and the relationships between structural elements of the glyph. Interpolates maintain alignment of and proportions between structural elements. While slight deviations of a glyph's knots from the grid are acceptable to the human eye, anchors help maintain the consistency across a string of glyphs. Delta and move exceptions are not translated as they are typically applied by the typographer on a case-by-case basis. Global-scope instructions (*Smooth()*, for example) are also not translated for individual glyphs. As such instructions typically apply to all glyphs, they can be applied separately in post-processing.

3.4 Stroke width regularization

Because each instruction is a local operation, hints alone cannot provide a typographer with complete control over consistency among glyphs. This additional expressive power is provided by the control value table (CVT), a shared table of distances referenced by hint instructions. References to entries in the CVT regularize the appearance of structural elements within a single glyph (e.g. when referenced by a distance instruction) or across glyphs (e.g. in the case of the link instruction) [Ado01]. Use of the CVT guarantees that values the typographer intended to be equal at design time are rendered as such.

It could be argued that the CVT is not appropriate for use with handwriting fonts because it introduces too much uniformity. We limit the restrictiveness of the CVT by tailoring it to the features of the input. As discussed in [Zon00], the CVT entry numbers of template can certainly still be used for our input. However, the values in these entries, designed for the particular features of the template, are no longer appropriate. We must calculate new values for the entries based on measured features of the input. We consider every instance where a specific CVT entry is referenced by template glyphs. We then average the actual values at analogous knots in the input glyphs to calculate the new CVT entry. Zongker et al. discarded as outliers those cases in which the measured value was too different from the average value. The reasoning is that the difference suggests that it is not appropriate to apply this CVT constraint in this case. While for uniform typefaces this approach results in relatively little loss of hint data, when considering handwriting, the wide variations found in measured values for a single CVT entry preclude use of this method.

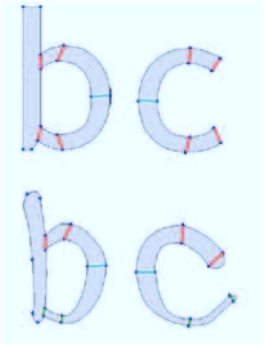

Figure 5. Identifying clusters of CVT references. Red lines indicate the templates□link references to the same CVT entry. After the hints are translated to the input glyphs, it becomes apparent that a new entry should be created for the cluster of green links.

We note that, due to the cross-letter patterns in a person's handwriting, these outliers often appear in clusters. While differing greatly from the average values stored in the CVT, these outliers are often close enough to each other to be considered a separate class of reference. An example is shown in Figure 5. Rather than discarding outliers, we partition references to a particular CVT entry into clusters of references. Sufficiently different references are branched into a new CVT entry. The averaging and branching continues until all entries have been categorized. An entry referenced by a single link instruction can safely be discarded and the link replaced with a distance instruction.

This clustering and branching approach allows us to identify patterns in the input set, retaining as much hint data as possible.

4 RESULTS AND DISCUSSION

Figure 6 shows a number of handwritten characters automatically hinted with our method. The input characters were manually segmented from complete words written on a tablet computer. A manually-hinted Roman font was used as the template for the global correspondence search; the local search used a hinted, stroke-like font as the template. We tested the results of autohinting glyphs displayed at a typical screen resolution of 96 dpi using Visual TrueType's internal rasterizer.

4.1 Hints and dropout control

A topic of ongoing discussion among typographers is whether italic fonts, fancy fonts and handwritten fonts need to be hinted or if for these fonts, only

basic hints and a *dropout control* mechanism are needed. When part of a stroke is thinner than one pixel, the resulting hole or "drop" in the raster image can be disruptive to perception of the character. To prevent these artifacts, a simple dropout control mechanism can be applied at time of rasterization to detect the location of drops and to insert an extra pixel at the site of the drop. (For an in depth discussion of dropout control, please see [Her93].)

In Figure 6, we compare glyphs with hints automatically applied, those with only dropout control applied, and those with both hints and dropout control applied.

As noted in Section 3.1, the handwritten glyphs contain no off-curve control points and a much larger number of on-curve control points. Because of this, the effect of the dropout control mechanism is to simply "connect the dots", resulting in a single-width polyline in many cases. (See, for example, the second 'c' at 18 pt in Figure 6.)

The automatically hinted glyphs show improvement in certain features at the cost of slight distortion of other features. (The 'm's in the figure are good examples of this.)

Combining auto-hinting and dropout control produces characters that are more legible that those using either mechanism alone and that are clearly a great improvement over unhinted characters.

Still, the matching algorithm is far from perfect; in some of the glyphs (e.g. the first 'b' at 18 pt, the first 'e' at 24 pt), the translation of inappropriate hints actually degraded the appearance. But while this and other automatic hinting systems still have a ways to go to come close to the hinting accuracy of expert typographers, these early results are encouraging.

4.2 Choosing templates

The choice of template, as well as the choice of whether to hint both globally and locally, depends on the purpose the hinted handwriting will serve. If the goal is to have consistently readable text, the best choice may be a professionally-hinted highly-uniform font template for global hinting only. If the goal is to provide the user with a "typographically nice" form of their writing, use of a large database of local templates will increase the likelihood of a close match. One could imagine using one automatically-hinted font as a template for another, but this would degrade the results.

4.3 Applications

Contextual handwriting fonts. The new OpenType standard, developed jointly by Adobe and Microsoft [Ado01], provides support for *contextual fonts* which can store multiple definitions of each glyph. Several typeface companies have already taken advantage of this technology in the handwriting fonts they produce. Typographers at Signature Software, Inc. use a semi-automatic system to design multiple forms for each cursive character so that each can connect naturally to one preceding. While the resulting fonts are more regularized than a person's actual handwriting, the contextually changing character connection locations help give the appearance that the person might have written the text. The techniques described in this paper make it feasible to automatically hint a large number of variations of each glyph for very realistic handwriting.

Hinting of arbitrary curves. Our hybrid correspondence search could be applied to discover structure in an arbitrary curve. We are interested in pursuing the extension to hinting of logos and vector graphics for optimal display on low resolution devices.

General rendering of handwriting. In this paper, we discussed example-based methods of improving rendering of handwriting in the context of TrueType font hinting. It would be worthwhile to consider the application of these techniques in a more general context, replacing the font templates and TrueType hints with a more general template and additional rendering information.

ACKNOWLEDGEMENTS

This project was initiated while S. Su and C. Wu were interns at Microsoft Research Asia, and we acknowledge our colleagues there, at the Microsoft Redmond campus, and in the MIT Computer Graphics Group for insightful discussions about this work. We also thank the anonymous reviewers for their feedback.

REFERENCES

[Ado01] Adobe Systems Inc., and Microsoft Corp. *OpenType Specification*, 1.3 ed., April 2001.

[Ado90] Adobe Systems Inc.. *Adobe Type 1 Font Format*. Addison-Wesley, 1990.

[App96] Apple Computer Inc. *The TrueType Reference Manual*. October 1996.

[Arv00] Arvo., J., and Novins, K. Smart Text: A synthesis of recognition and morphing. In *Proc. of AAAI Spring Symposium on Smart Graphics*, pp. 140-147, 2000.

[Bra91] Branston, B. *Graphology Explained*. Samuel Weiser Inc., 1991.

[Con97] Connare, V. *Basic Hinting Philosophies and TrueType Instructions*, Microsoft Corporation, 1997.

[Dev95] Devroye, L., and McDougall, M. Random fonts for the simulation of handwriting. *Electronic Publishing*, Vol. 8, pp. 281-294, 1995.

[Guy96] Guyon, I. Handwriting synthesis from handwritten glyphs. In *Proc. of the 5th International*

Workshop on Frontiers of Handwriting Recognition, 1995.

[Her91] Hersch, R.D., and Bétrisey, C. Model-based matching and hinting of fonts. In *Proc. of SIGGRAPH 91*, pp. 71-80, 1991.

[Her93] Hersch, R. Font rasterization: the state of the art. In *Visual and Technical Aspects of Type*, R. Hersch (ed.), Cambridge University Press, pp. 78-109, 1993.

[Her94] Hertz, J., and Hersch, R.D. Towards a universal autohinting system for typographic shapes. *Electronic Publishing*, Vol. 7, pp. 251-260, December 1994.

[Hob93] Hobby, J.D. Generating automatically tuned bitmaps from outlines. *Journal of the ACM*, Vol. 40, No. 1, pp. 48-94, 1993.

[Kla93] Klassen, R.V. Variable width splines: a possible font representation? *Electronic Publishing*, Vol. 6, No. 3, pp. 183-194, September 1993.

[Knu86] Knuth, D.E. *The* METAFONT *Book*. Addison-Wesley, 1986.

[Mac94] MacKenzie, I.S., Nonnecke, B., McQueen, C., Riddersma, S., and Meltz. M. Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies.* Vol. 41, pp. 775-792.

[Mac97] MacKenzie, I.S., and Zhang, S. The immediate usability of Graffiti. In *Proc. of Graphics Interface '97.* pp. 129-137, 1997.

[McG95] McGraw, G.E. *Letter Spirit: Emergent High-Level Perception of Letters Using Fluid Concepts*. PhD thesis, Indiana University, 1995.

[Rub88] Rubinstein, R. *Digital Typography: An Introduction to Type and Composition for Computer System Design.* Addison-Wesley, 1988.

[Sha03] Shamir, A. Constraint based approach for automatic hinting of digital typefaces. *ACM Transactions on Graphics*, Vol. 22, No. 2, April 2003.

[Sta97] Stamm, B. *The Raster Tragedy at Low Resolution*. Microsoft Corporation, 1997.

[Sta98] Stamm, B. Visual TrueType: a graphical method for authoring font intelligence. In *Proc. of Raster Imaging and Digital Typography '98*, pp. 77-92, 1998.

[Sut63] Sutherland, I.E. *Sketchpad: A Man-Machine Graphical Communication System.* PhD thesis, Massachusetts Institute of Technology, 1963.

[Typ96] TYPE*chimérique Organization. *TrueType Hinting*. 1996.

[Wan02] Wang, J., Wu., C., Xu, Y.-Q., Shum, H.-Y., and Ji., L. Learning-based cursive handwriting synthesis. In *Proc. of the 8th International Workshop on Frontiers in Handwriting Recognition,* 2002.

[Zon00] Zongker, D.E., Wade, G., and Salesin, D.H. Example-based hinting of TrueType fonts. In *Proc. of SIGGRAPH 2000*, 2000.

Figure 6. Comparison of glyphs without hints, with hints automatically applied, with only dropout control applied, and with both hints and dropout control applied. Manually-hinted Roman and stroke-like glyphs were used as the templates for the global and local correspondence searches, respectively. Results are shown at typical screen resolution of 96 dpi.