Classification of TrueType Font Using Clustering Region

Seongah CHIN and Moonwon CHOO
Department of Multimedia
Sungkyul University, KOREA
#147-2, Anyang 8-dong, Manan-gu, Anyang-city, Kyunggi-do, KOREA
schin@soback.kornet.net,mchoo@hana.sungkyul.ac.kr

Abstract: As we review the mechanism regarding digital font generation and birth of TrueType font, we realizes that the process is composed of sequential steps such as contour fonts from glyph table. This fact implies that we propose classification of TrueType font in terms of segment width and the number of occurrence from the glyph data.

1. Introduction

The historical development of the TrueType font system was originally designed by Apple Computer Inc.. We discover the history of digital TrueType font system in terms of origin of birth, developed procedures and historical background. TrueType is the scalable font technology built into Windows and Macintosh. Apple had been developing what was to become TrueType from late 1987. A lead engineer, Sampo Kaasila completed his work on TrueType in August 1989. Apple included full TrueType support in its Macintosh operating System 7, in May 1990. Microsoft first included TrueType into Windows 3.1 in April 1991. It supported TrueType version of Times Roman, Arial and Courier, which was same as Apple's [1] [2]. One similar work suggested based on typographical features is related to optical font recognition [3].

Our intention is to explore the purpose of the font density function over segment widths, which makes it possible to classify TrueType font. As we look into True Time font, each font sketches a unique shape consisting of smooth curves and straight lines. A smooth curve is defined by quadratic B spline curves, which can segment into several quadratic Bezier curves holding three control points. A straight line can be drawn by two end points. Mathematically, we can derive and define the segment width function, denoted by W, which is the shortest distance between intersection points with segments of individual fonts on a horizontal scan. If it is known each height is distributed uniformly, we can apply the fundamental theorem to obtain our density function in terms of uniformly distributed height h and segment width function W.

Finally, this approach makes the clustering region solid and durable so that we can point out the exact position of the region where the specific font lies. In order to verify our idea, we will show the numerical approaches.

2. Theoretical View of Font Density Function

The main goal of this section is to describe theoretical approaches to derive the font density function analytically. This section utilizes the fact that the boundary function of some scalable fonts (e.g. TrueType [2]) consists of combinations of Bezier curves or straight lines. Therefore the width W, of a typeface may be determined analytically. Hence we will specify mathematical definitions of the segment width W and its density function.

2.1 TrueType Font System

This section brings in the key concepts how the font system works. The TrueType Font System is thought of as the engine converting the information in a TrueType font into a raster image suitable for display on screen or printer.

Prior to giving the explanation of the whole procedure, we supply a glyph table, which contains the data (x,y coordinate data) of the appearance of the glyphs in the fonts as shown in **Fig. 1**. Each glyph in a TrueType font is described by a sequence of points on a grid. While two on curve points are sufficient to describe a straight line, the addition of a third off curve point between two on curve points makes a parabolic curve. On curve points marked by '·' and off curve points by 'o' in **Fig. 1**.

The following steps illustrate the sequential processes for displaying TrueType characters on raster device [4].

- Scaling the outline information of glyph to the requested size
- Grid fitting into scaled outline based on associated instruction
- Decision of inside and outside by scan converter
- Rendering a bitmap image suitable for raster display

2.2 Boundary Generation

TrueType curves are defined by quadratic B-splines, which are composed of a series of quadratic Bezier curves. Three control points of which the first and the last are on curve points and the middle one is the off curve point define these curves. The off curve point is located at the intersection point between two slopes of the curve at the first and last points. Mathematically, the following expressions give general parametric equation [10],[11].

$$P_x(t) = \sum_{i=0}^{n} C_i^n t^i (1-t)^{n-i} \alpha_{x_i} \quad (2.1)$$

$$P_y(t) = \sum_{i=0}^{n} C_i^n t^i (1-t)^{n-i} \alpha_{y_i} \quad (2.2)$$

Where parameter t is such that $0 \le t \le 1$ and $P_x(t)$ is the x value and $P_y(t)$ is the y value and α_{xi} and α_{yi} are real number defining the shape of the curve segment.

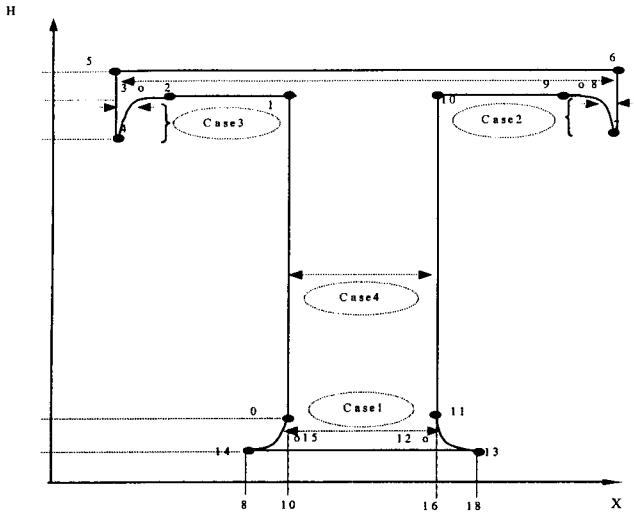

Fig. 1.
Segment width types and Glyph Data

2.3 TrueType File Formats

On Microsoft Window platforms (Windows 95, Windows 98, Windows NT) TrueType font definition files are stored in the Font subdirectory within the operating system directory (usually WINDOWS or WinNT). The files have extension ".ttf". Any of these TrueType files may read or viewed using the utility TTFDump. TTFDump is a command line tool that dumps the contents of TrueType font files. TTFDump parses and labels the contents of the tables. Using TTFDump we can extract tables and data for specific glyphs

2.4 Font Segment Width Function W

The purpose of this section is to define the segment width in terms of Bezier curves and straight lines in the font coordinate system, where x goes horizontally and y increases vertically. Given a y position, we will endeavor to find out intersection points with Bezier curves and straight lines so that we can define the segment width W as the function of y (denoted by h : height from this moment).

Notational Adjustment

We accounts for notations for the definition of segment width W. A segment width can be determined by a combination of the pair of Bezier curves and straight lines. Thus, we need to adjust the notation to develop the approaches as follows:

- Bezier curve on left side denoted by B_l
- Bezier curves on right side marked by B_r
- Straight line on left side promised by L_l
- Straight line on right side called by L_r
- Parameter t associated with left curve or line denoted by t_l
- Parameter t associated with right curve or line denoted by t_r
- y coordinate denoted by h

Now let us define W in terms of h (same as y). Given h, we note there exist four cases when a scan segment occurs as follows:

Case 1 : (B_l, B_r) i.e both sides of a segment are quadratic Bezier curves
Case 2 : (B_l, L_r) i.e the left side is a quadratic Bezier curve and the right side is a straight line
Case 3 : (L_l, B_r) i.e the left side is a straight line and the right side is a Bezier curve
Case 4 : (L_l, L_r) i.e both are straight lines

An example of each segment case from the letter 'T' is shown in **Fig. 1.** We are interested in deriving analytic expressions for the segment width, W, as a function of the height, h. i.e. $W = g(h)$

Definition of W for Case 1 : (B_l, B_r)
We begin with Case 1 where the left Bezier curve B_l associates with three control points marked by (A_x, A_y), (B_x, B_y), (C_x, C_y) and B_r defined by (A'_x, A'_y), (B'_x, B'_y), (C'_x, C'_y) There are four defining equations this problem :

$$P_y(t_l) = (1-t_l)^2 A_y + 2t_l(1-t_l)B_y + t_l^2 C_y \quad (2.1)$$

$$P_x(t_l) = (1-t_l)^2 A_x + 2t_l(1-t_l)B_x + t_l^2 C_x \quad (2.2)$$

$$P_y(t_r) = (1-t_r)^2 A'_y + 2t_r(1-t_r)B'_y + t_r^2 C'_y \quad (2.3)$$

$$P_x(t_r) = (1-t_r)^2 A'_x + 2t_r(1-t_r)B'_x + t_r^2 C'_x \quad (2.4)$$

We are able to obtain the parameter variable t_l applying A_y, B_y, C_y into the equation and substituting $P_y(t_l)$ with h in equation 2.1 as follows:

$$h = (1-t_l)^2 A_y + 2t_l(1-t_l)B_y + t_l^2 C_y$$

Then solve t_l using quadratic solution and rewrite it as :

$$t_l = \alpha_y \pm \beta_y \sqrt{\chi_y + h} \quad (2.5)$$

Where

$$\alpha_y = \frac{(A_y - B_y)}{(A_y - 2B_y + C_y)}$$

$$\beta_y = \frac{\sqrt{A_y - 2B_y + C_y}}{(A_y - 2B_y + C_y)}$$

$$\chi_y = \frac{(B_y - A_y)^2}{(A_y - 2B_y + C_y)} - A_y$$

Similar manipulation on equation 2.2 gives :

$$t_l = \alpha_x \pm \beta_x \sqrt{\chi_x + P_x(t_l)} \quad (2.6)$$

Where

$$\alpha_x = \frac{(A_x - B_x)}{(A_x - 2B_x + C_x)}$$

$$\beta_x = \frac{\sqrt{A_x - 2B_x + C_x}}{(A_x - 2B_x + C_x)}$$

$$\chi_x = \frac{(B_y - A_y)^2}{(A_x - 2B_x + C_x)} - A_y$$

From (2.5) and (2.6) we can merge both equations and eliminate t_l.(as mentioned before) into :

$$\alpha_y - \alpha_x \pm \beta_y \sqrt{\chi_y + h} = \pm \beta_x \sqrt{\chi_x + P_x(t_l)} \quad (2.7)$$

Similarly, the expression for t_r, will be reduced to :

$$t_r = \alpha'_y \pm \beta'_y \sqrt{\chi'_y + h} \quad (2.8)$$

From **Fig. 2.** we see that $P_x(t_r) = P_x(t_l) + W$
Therefore substituting this in equation 2.4 we obtain :

$$t_r = \alpha'_x \pm \beta'_x \sqrt{\chi'_x + P_x(t_l) + W} \quad (2.9)$$

Where

$$\alpha'_y = \frac{A'_y - B'_y}{A'_y - 2B'_y + C'_y}$$

$$\beta'_y = \frac{\sqrt{A'_y - 2B'_y + C'_y}}{A'_y - 2B'_y + C'_y}$$

$$\chi'_y = \frac{(B'_y - A'_y)^2}{A'_y - 2B'_y + C'_y} - A'_y$$

$$\alpha'_x = \frac{A'_x - B'_x}{A'_x - 2B'_x + C'_x}$$

$$\beta'_x = \frac{\sqrt{A'_x - 2B'_x + C'_x}}{\sqrt{A'_x - 2B'_x + C'_x}}$$

$$\chi'_x = \frac{(B'_x - A'_x)^2}{A'_x - 2B'_x + C'_x} - A'_x$$

By equating equation (2.8) and (2.9) we eliminate t_r (as precisely stated) resulting in :

$$\alpha'_y - \alpha'_x \pm \beta'_y \sqrt{\chi'_y + h} = \pm \beta'_x \sqrt{\chi'_x + P_x(t_l) + W} \quad (2.10)$$

From the equation (2.7) and (2.10), we obtain :

$$\left(\frac{(\alpha_y - \alpha_x)}{\beta_x} \pm \frac{\beta_y}{\beta_x} \sqrt{\chi_y + h} \right)^2 = \chi_x + P_x(t_l) \quad (2.11)$$

$$\left(\frac{(\alpha'_y - \alpha'_x)}{\beta'_x} \pm \frac{\beta'_y}{\beta'_x} \sqrt{\chi'_y + h} \right)^2 = \chi'_x + P_x(t_l) + W \quad (2.12)$$

Subtracting (2.11) from (2.12) will eliminate $P_x(t_l)$ leaving us with an expression relating h and W

$$(a' \pm b' \sqrt{\chi'_y + h})^2 - (a \pm b \sqrt{\chi_y + h})^2 = \chi'_x - \chi_x + W \quad (2.13)$$

Where

$$a' = \frac{(\alpha'_y - \alpha'_x)}{\beta'_x}, \qquad b' = \frac{\beta'_y}{\beta'_x}$$

$$a = \frac{(\alpha_y - \alpha_x)}{\beta_x}, \qquad b = \frac{\beta_y}{\beta_x}$$

Finally, the functional form of W appears as below,

$$W = (a' \pm b' \sqrt{\chi'_y + h})^2 - (a \pm b \sqrt{\chi_y + h})^2 - \chi'_x + \chi_x \quad (2.14)$$

Now, let W be $g(h)$ which is a complex function of its argument h,

$$W = g(h) \quad (2.15)$$

Equation (2.14) allows us to compute the width, or separation, between two Bezier curves along a horizontal line. If the two Bezier curves are the boundaries of typeface of a character then this equation gives us an analytic expression for the width of the character as a function of its height. An inverse function is not easy to derive and numerical methods, such as Newton's [12], may be used to obtain the height if the character width is given. Likewise, we can derive the other cases for Case2, Case3 ands Case4.

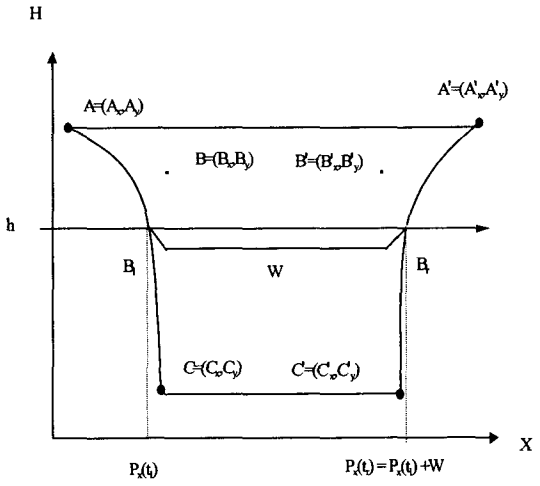

Fig. 2. Segment width W and its coordinate system when both sides are Bezier curves, B_l and B_r

2.3 Font Density Function

We develop the font density function based on the segment width function W from the previous section using the fundamental theorem for the transformation of random variables [13]. The rationale of for modeling the segment width, w, as a random variables arises from the fact a "randomly" positioned scan line through an image (containing text) may intersect a text character and produce a segment width of "random" length . Of course the widths are not total random because they depend on a) the specific letter encountered and b) the position on the letter where the scan line crosses the letter. If we model the scan position or height-h, as a uniformly distributed random variable (**Fig. 4**) then the related segment width , w, is a random variable with the density (**Fig. 5**) . When the shape

or typeface of the letter changes then the resulting density functions changes. Consider the shape shown in **Fig. 3** Scan line1 produces three line segments, two of width w and one with a width between 0 and W. Since some scan lines produce more than a single segment width the density of the height random variable is adjusted to accommodate this. In this example,

$$f(h)=\frac{1}{H}(u(h)-u(h-H)) \; ; H = H_1 + H_2 + 2H_3 \qquad (2.16)$$

Where u(.) is the unit step function and H is the total height of letter T. The probability density of the segment width will be a mixed density since w will a discrete and continuous random variable. The wedged serif of two T contributes to continuous part. The density is

$$f_w(w|T)=\frac{H_2+H_3}{H}\delta(w-w_2)+\frac{H_1}{H}\delta(w-w_1)+\frac{H_3}{Hw}(u(w)-u(w-w_3)) \qquad (2.17)$$

In general the height random variable , h, is converted by a transformation, T, which is font dependent, into a segment width random variable, w. Since scalable fonts are mathematically defined the transformation for these fonts is also (possibly) definable and therefore the probability density (in theory) of w is desirable. In practice the inverse function, T^{-1} is required to express the density an in many case the result are not tractable. We now proceed with general derivation of density function of segment width given a specific letter and a specific font.

As we saw in the previous section, the segment width W can be fixed, in special case when both sides are line segments and the slopes of the line segment are same (i.e Case 4s). The other cases Case 1, Case 2 and Case 3, and Case 4 represents situations where continuous segment widths W where W increases or decreases as h varies. Thus, segment density function will be defined as consisting of two classes : continuous and fixed.

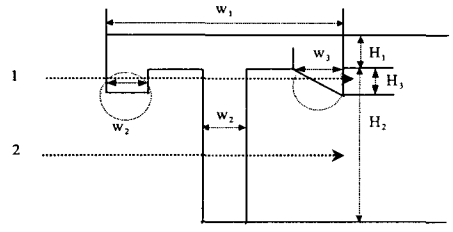

Fig. 3.

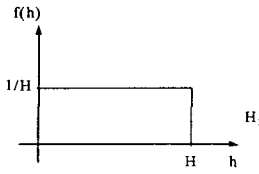

Fig. 4.

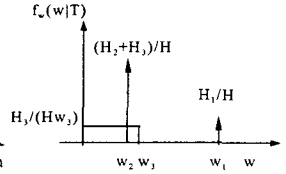

Fig. 5.

Class 1 : continuous segment widths W
Class 2 : fixed segment widths W

$$Class\ 1 : f_w(w_i|L)=\begin{cases}\dfrac{f_h(h)}{\left|\dfrac{d}{dh}g(h)\right|} & ; \; w_i \in \text{Class1} \;\; h \in h_j \\ 0 & ; \; \text{otherwise}\end{cases}$$

$$Class2 : \quad f_w(w_i|L)=\begin{cases}\dfrac{h_i}{H}\delta(w_i-c) & ; w_i \in \text{Class2}, w_i=c, \; H: \text{total height} \\ & h_i \text{ is height of fixedwidth} w_i \\ 0 & ; \; \text{otherwise}\end{cases}$$

Where $f_h(h) = 1/H, \; h \in h_j$, associated with $w_i \in$ Class 1 with $H = \sum_{j=1}^{N} h_j$ for $w_I \in$ Class 2.

In **Fig. 1**, we can distinguish continuous segment widths, (Class1) from fixed segment widths Class 2 and some of Case 4. In other words, Class1 is grouped with Case1, Case2, Case3 and Case 4, while Case4s is belong to Class2 in **Fig. 1**. But Case 1 can be Class1 when segment width keeps fixed width. In addition, Case 4 probably can be classified as Class1 when segment width varies on heights *h*. We note a total height *H* can be defined the total number of heights assosiated with segment width w_I. The width density function of a font , denoted by *f(w)*, can be defined based on the fact that some letters frequently appear in a English sentence, whereas others seldom show up. In other words, each individual letter has its own probability.

$$f(w) = \sum_{L=A,B,\ldots,Z} P(L=j) f_l(w \mid L=j) \qquad (2.18)$$

Where *P(L)* is the probability of a letter in an English sentence and *f(w|L)* is the segment width density function.

3. Experiments

We have learned that TrueType font definition files are stored in the Font subdirectory within the operating system directory . We are able to extract glyph data using the utility TTFDump that dumps the contents of TrueType font files. TTFDump parses and labels the contents of the tables. Using TTFDump we can extract tables and data for specific glyphs. TTFDump may be download from the following web site :

http://www.microsoft.com/typography/tools/tools.htm

Example : Dumping all tables of Times Roman (times.ttf) to a text file (times.txt)

C:\>ttfdump times.ttf > times.txt

Example : Dumping all glyph data for Times Roman to a text file (glyph.txt)

C:\>ttfdump times.ttf -nx -tglyf> glyph.txt

Since an inverse function is not easy to derive so that a histogram over the segment width provides the font density distribution for that specific letter. Combining the

Table 1. Times Roman Upper Case Glyph information

Letter	Scan line count			Maximal width		2nd Maximal	
width	x count	y count	width segments	size	count	size	count
A	1441	1388	2195	203	464	96	330
B	1221	1357	1428	211	86	226	30
C	1222	1419	4085	70	105	226	26
D	1367	1357	1310	224	104	228	69
E	1163	1357	2417	193	620	741	73
F	1023	1357	3035	193	186	172	82
G	1379	1418	3138	196	428	236	118
H	1403	1357	2628	193	1288	582	75
I	582	1357	1357	193	911	581	40
J	744	1388	1703	193	698	194	23
K	1463	1356	2919	193	662	267	210
L	1167	1357	1873	193	772	615	39
M	1745	1357	4913	89	939	193	906
N	1478	1378	4494	89	1653	45	285
O	1329	1419	2590	230	231	229	133
P	1034	1357	1419	193	418	226	87
Q	1330	1789	1419	193	418	226	87
R	1350	1357	2958	240	246	241	138
S	902	1419	2831	37	27	45	26
T	1139	1357	1872	193	867	576	37
U	1447	1389	3115	193	569	91	594
V	1437	1388	2640	207	786	93	448
W	1891	1388	4682	88	926	98	422
X	1441	1357	2852	225	137	106	125
Y	1431	1357	2152	98	490	193	323
Z	1169	1357	1898	227,228,229	296	230	190

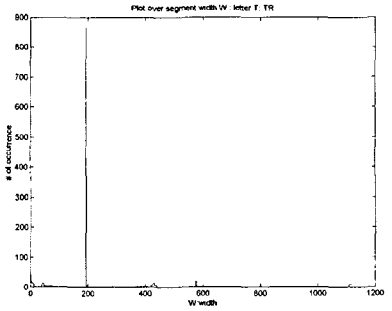

Fig. 6. Histogram of 'T' in Times Roman

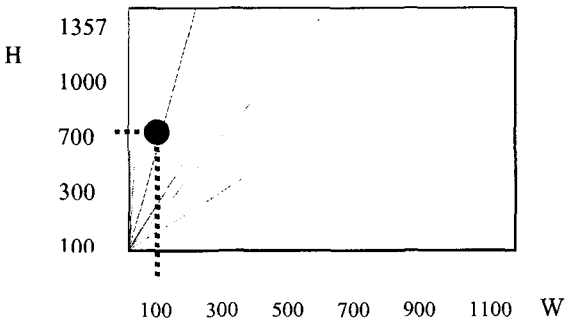

Fig. 7. The Clustering Region R

histogram data and the probabilities associated with individual letters, we obtain the font density distribution which specify the clustering region. The region in Height/Width space where this density exceeds some threshold (τ_R) is the Clustering Region.

$$\mathbf{R} = \{(W, H) | f(W, H) > \tau_R\} \quad (2.19)$$

Table 1 shows statistical data for the Times Roman uppercase glyph information. We note that higher numbers of occurrences are found in scan line widths around 200. In **Fig. 6**, we can see high appearance in scan line width at 200 for Times Roman "T". **Fig. 7** displays clustering region ***R*** in terms of height over width space.

4. Conclusion

We propose classification methods of TrueType font based on investigating glyph information. This results in providing sufficient features in terms of font density function since typeface is defined by its boundary on curve points or off curve points. This approach makes it possible to point out the exact position of the region where the specific font lies building clustering region in scan line width over height space. This work can extend to recognize various typefaces called font recognition system as future work.

References

[1] Laurence Penny, "A history of TrueType," TrueType Typography Technical report, TrueType Development Team at Apple, 1999

[2] L Penny, "A brief history of TrueType," A article on the TrueType from interview with Kassila, The Principal Inventor, June, 1997

[1] A. Zremdini, R. Ingold, "Optical Font Recognition Using Typographical Features," IEEE Transaction on Pattern Analysis And Machine Intelligence. Vol. 20. No. 8, pp. 877-882, August 1998

[2] D Herman and Apple TrueType team, "The TrueType Reference Manual,", Apple Computer Inc. Feb, 1998

[3] "Microsoft Typography Quick Reference Guide V.1.0," http://www.miscroft.com/typography/tools/tools.htm

[4] Donald E. Knuth, Digital Typography, Center for the Study of language and information Stanford Junior University, 1999

[5] Donald E. Knuth, Tex & METAFONT New direction on Typesetting American Mathematics Society 1979

[6] S-H. F. Chuang, · C.Z Kao, "One-Sided arc approximation of B-spline curves for interference – free offsetting," Computer-Aided Design 31(1999) pp. 111-118, 1999

[7] Kamran Etemad, David Doermann, and Rama Chellappa, "Multiscale Segmentation of unstructured Document Pages Using Soft Decision Integration,"

[8] R. H. Bartels, An Introduction to Spline for use in computer Graphics & Geometric Modeling, Morgan Kaufmann Publishers Inc. 1987

[9] R. C. Beach, An Introduction to the curves and surface of Computer-Aided Design, Van Nostrand Reinhold 1991

[10] M.L. James/G.M. Smith. J.C. Wolford, "Applied Numerical Method for Digital Computation, Harper & Row, Publishers, 1985

[11] A. Papoulis, "Probability, Random Variables and Stochastic Processes", Third Edition, McGraw-Hill, Inc. pp. 92-102, 1991