워드이미지로부터 영문인식을 위한 트루타입 특성 추출

진성아*

Deriving TrueType Features for Letter Recognition in Word Images

SeongAh CHIN

Abstract

In the work presented here, we describe a method to extract TrueType features for supporting letter recognition. Even if variously existing document processing techniques have been challenged, almost few methods are capable of recognize a letter associated with its TrueType features supporting OCR free, which boost up fast processing time for image text retrieval. By reviewing the mechanism generating digital fonts and birth of TrueType, we realize that each TrueType is drawn by its contour of the glyph table. Hence, we are capable of deriving the segment with density for a letter with a specific TrueType, defined by the number of occurrence over a segment width. A certain number of occurrence appears frequently often due to the fixed segment width. We utilize letter recognition by comparing TrueType feature library of a letter with that from input word images. Experiments have been carried out to justify robustness of the proposed method showing acceptable results.

Key Words: Font, Document Processing, OCR, Text

* 성결대학교 멀티미디어학부

1. Introduction

Ironically, more paper documents are being produced while electronic documents are constantly increasing. This fact draws that we constantly need to develop document processing providing more conveniently acquiring textual information from broad sources including Web images, micro films, digital images and paper documents. Text information from those images makes the search engines more flexible and intelligent. In order to identify each letter, a general method is to convert segmented textual area into OCR generating a machine understandable code format. Prior to making code format of an individual letter, it is better to recognize directly without coding stage. Thus, in the research presented we derive TrueType features for letter recognition without OCR stage. The fundamental idea is derived from learning characteristics of TrueType fonts holding the properties that a certain number of occurrence appears frequently often due to dominant segment width, denoted by W, which is the shortest distance between intersection points with segments of individual fonts on a horizontal scan. 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 letter density function over segment widths, which makes it possible to identify letters associating with a 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 builds TrueType feature library of letters associating with a font capable of recognition of input word images by comparing features. In order to verify our idea, we will show the numerical approaches by experiments. We utilize letter

recognition by comparing TrueType feature library of a letter with that from input word images. Experiments have been carried out to justify robustness of the proposed method showing acceptable results.

2. Previously Related Work

The history of digital TrueType font system in terms of origin of birth developed procedures and historical background is introduced here followed by introducing previously related research work. TrueType is the scalable font technology built into Windows and Macintosh. Back in the late 1980s, it was clear to most of the major players in the personal computer world that scalable font technology was going to be an important part of the future operating system. Adobe was trying to get Apple and Microsoft to license its PostScript code for this purpose. However, both companies were naturally concerned about giving control over key parts of their operating systems to Adobe. 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 and Microsoft announced their strategic alliance against Adobe, where Apple would design the font system [15]. 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 [16]. Previously conducted methods have been developed by Z. - Yu [20]. His idea is to classify character objects in the document called Vertical Traverse Density and Horizontal Traverse Density followed by constructing n-gram vector and computing similarity between pair of documents. However, his method may not notice font and its size whereas our proposed scheme aims at recognizing font and size as well as letter. Chen F.R [21] proposed a text summarization method without the use of OCR supporting only one predominant font and language dependent. In addition, A. Zramdini and yong Zhu [13][19] rather focused on identifying fonts than recognizing individual letters. Zhu employs multichannel Gabor filtering technique for feature extraction followed by applying these feature into weighted Euclidean distance classifier for deriving font identification. However, our method is originally designed and developed to detect not only font and size but also each letter.

3. TrueType Features

3.1 TrueType Font System

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 Figure. 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 '' in Figure 1. In a TrueType font, glyph shapes are described by their outlines. A glyph outline consistsof a series of contours. A simple glyph may have only one contour. More complex glyphs can have two or more contours. Composite glyphs can be constructed by combining two or more simpler glyphs. Certain control characters that have no visible manifestation will map to the glyph with no contours. Contours are composed of straight lines and curves. Curves are defined by a series of points that describe second order Bezier-splines. The TrueType Bezier format uses two types of points to define curves, those that are on the curve and those that are off the curve. Any combination of off and on curve points is acceptable when defining a curve. Straight lines are defined by two consecutive on curve points.

3.2 Theoretical View of letter density Function

The main goal of this section is to describe theoretical approaches to derive the letter 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. 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].

$$P_x(t) = \sum_{i=0}^{n} C_i^n t^i (1-t)^{n-i} \alpha_{x_i} \quad 3\text{-}1$$

$$P_y(t) = \sum_{i=0}^{n} C_i^n t^i (1-t)^{n-i} \alpha_{y_i} \quad 3\text{-}2$$

Where parameter t is such that 0 t 1 and Px(t) is the x value and Py(t) is the y value and xi and yi are real number defining the shape of the curve segment.

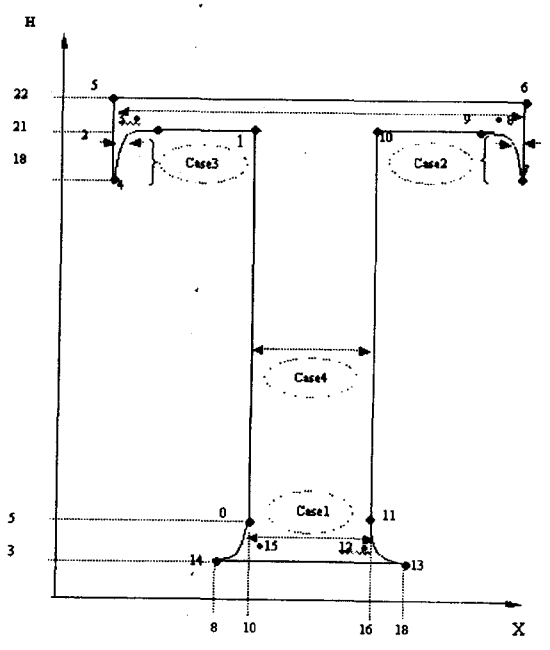

Figure 1. Glyph Table coordinate data and various kinds of segment width

3.3 Letter 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 Bl, Bezier curves on right side marked by Br, Straight line on left side promised by Ll, Straight line on right side called by Lr, Parameter t associated with left curve or line denoted by tl, Parameter t associated with right curve or line denoted by tr 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:

Case1:(Bl, Br) i.e both sides of a segment are quadratic Bezier curves
Case 2 :(Bl, Lr) i.e the left side is a quadratic Bezier curve and the right side is a straight line
Case3 : (Ll, Br) i.e the left side is a straight line and the right side is a Bezier curve
Case4 : (Ll, Lr) i.e both are straight lines

An example of each segment case from the letter 'T' is shown in Figure 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 : (Bl, Br)

We begin with Case 1 where the left Bezier curve Bl associates with three control points marked by (Ax, Ay), (Bx, By), (Cx, Cy) and Br defined by (A'x, A'y), (B'x, B'y), (C'x, C'y). There are four defining equations this problem :

$$P_y(t_l) = (1-t_l)^2 A_y + 2t_l(1-t_l)B_y + t_l^2 C_y \quad 3\text{-}3$$

$$P_x(t_l) = (1-t_l)^2 A_x + 2t_l(1-t_l)B_x + t_l^2 C_x \quad 3\text{-}4$$

$$P_y(t_r) = (1-t_r)^2 A'_y + 2t_r(1-t_r)B'_y + t_r^2 C'_y \quad 3\text{-}5$$

$$P_x(t_r) = (1-t_r)^2 A'_x + 2t_r(1-t_r)B'_x + t_r^2 C'_x \quad 3\text{-}6$$

We are able to obtain the parameter variable tl applying Ay, By, Cy into the equation and substituting Py(tl) with h in equation 3-3. as follows:

$$h = (1-t_l)^2 A_y + 2t_l(1-t_l)B_y + t_l^2 C_y \quad 3\text{-}7$$

Then solve tl using quadratic solution and rewrite it as :

$$t_l = \alpha_y \pm \beta_y \sqrt{\chi_y + h} \quad 3\text{-}8$$

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 3-4, 3-5, and 3-6 give :

$$t_l = \alpha_x \pm \beta_x \sqrt{\chi_x + P_x(t_l)} \quad 3\text{-}9$$

$$t_r = \alpha'_y \pm \beta'_y \sqrt{\chi'_y + h} \quad 3\text{-}10$$

$$t_r = \alpha'_x \pm \beta'_x \sqrt{\chi'_x + P_x(t_l) + W} \quad 3\text{-}11$$

From Figure. 2. we see that $Px(tr) = Px(tl) + W$. Therefore substituting this in equation 3-6 we obtain the equation 3-11.

From 3-8 and 3-9 we can merge both equations and eliminate tl.(as mentioned before) into :

$$\alpha_y - \alpha_x \pm \beta_y \sqrt{\chi_y + h} = \pm \beta_x \sqrt{\chi_x + P_x(t_l)} \quad 3\text{-}12$$

By equation 3-10 and 3-11 we eliminate tr (as precisely stated) resulting in :

$$\alpha'_y - \alpha'_x \pm \beta'_y \sqrt{\chi'_y + h} = \pm \beta'_x \sqrt{\chi'_x + P_x(t_l) + W} \quad 3\text{-}13$$

From the equation 3-12 and 3-13, we obtain :

$$\left({}^{(\alpha_y - \alpha_x)}/_{\beta_x} \pm \frac{\beta_y}{\beta_x \sqrt{\chi_y + h}} \right)^2 = \chi_x + P_x(t_l) \quad 3\text{-}14$$

$$\left({}^{(\alpha'_y - \alpha'_x)}/_{\beta'_x} \pm \frac{\beta'_y}{\beta'_x \sqrt{\chi'_y + h}} \right)^2 = \chi'_x + P_x(t_l) + W \quad 3\text{-}15$$

Subtracting 3-14 from 3-15 will eliminate Px(tl) leaving us with an expression relating h and W

$$(a' \pm b' \sqrt{\chi'_y + h})^2 - (a \pm b \sqrt{\chi_y + h})^2 = \chi'_x - \chi_x + W \quad 3\text{-}16$$

Where

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

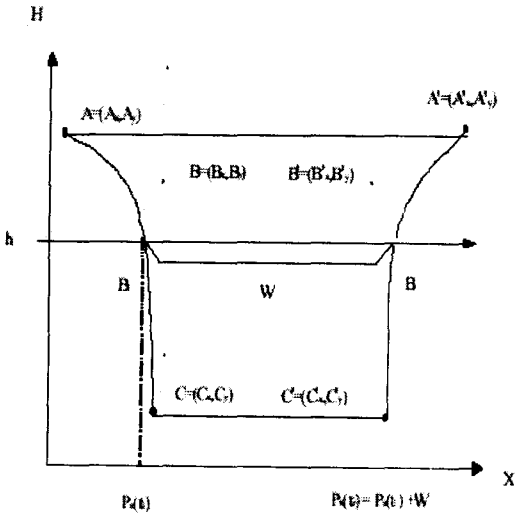

Figure 2. Segment width W and its coordinate system when both sides are Bezier curves, Bl and Br

Finally, the functional form of W appears as below,

$$W = (a' \pm b' \sqrt{\chi'_y + h})^2 - (a \pm b \sqrt{\chi_y + h})^2 - \chi'_x + \chi_x \quad 3\text{-}17$$

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

$$W = g(h) \quad 3\text{-}18$$

Equation 3-18 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 [10], may be used to obtain the height if the character width is given. Likewise, we can derive the other cases for Case2, Case3 and Case4.

3.4 Letter Density Function

We develop the letter density function based on the segment width function W from the previous section using the fundamental theorem for the transformation of random variables [12]. 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 totally 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 (Figure 3) then the related segment width, w is a random

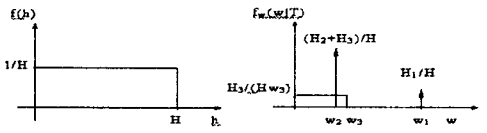

Figure 3. The related segment width w is a random variable with the density

variable with the density (Figure 3). Consider the shape shown in Figure 4 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.

$$f(h)=\frac{1}{H}(u(h)-u(h-H));H=H_1+H_2+2H_3 \qquad 3\text{-}19$$

In this example, 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

$$f_w(w|T)=\frac{H_2+H_3}{H}\delta(w-w_2)+\frac{H_1}{H}\delta(w-w_1)+\frac{H_3}{Hw}(u(w)-u(w-w_3)) \qquad 3\text{-}20$$

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

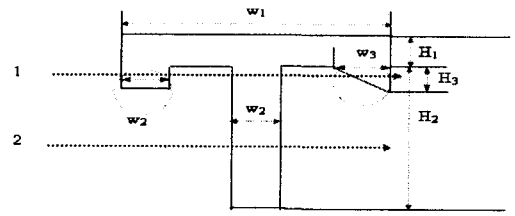

Figure 4. Model the scan position or height-h, as a uniformly distributed random variable

$$Class1 : f_1(w_i|L) = \begin{cases} \dfrac{f_1(h)}{\left|\dfrac{d}{dh}g(h)\right|} & ; \ w_i \in \text{Class1 } h \in h_j \\ 0 & ; \ \text{otherwise} \end{cases}$$

$$Class2 \ f_2(w_i|L) = \begin{cases} \dfrac{h_w}{H}\delta(w_i - c) & ; w_i \in \text{Class2}, w_i = c, \text{ H : total height} \\ & h_w \text{ is height of fixedwidthw,} \\ 0 & ; \ \text{otherwise} \end{cases}$$

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-1is 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 represent 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.

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

Fundamental Theorem of TrueType Features

Where fh(h) = 1/H, h hj, associated with wi Class 1 with $H = \sum_{j=1}^{N} h_j$ for wI Class 2. In Figure 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 Case4 is belong to Class2 in Figure 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 associated with segment width w. The letter density function of a font , denoted by f(w|L), can be defined as

$$f(w|L) = \sum_{i=1}^{n} f(wi|L) \qquad 3\text{-}21$$

Where i is the index of the segment width and f(w|L) is the segment width density function associated with a letter L.

4. Experiments and Discussion

In order to experiment font features described in section 3, we focuses on describing a numerical method to determine the letter density function of the segment width W. A glyph table provides each fonts' glyph information, which is the input data for our method. Each letter of the font has its own glyph data which define the shape boundaries of the typeface and consist of Bezier curves and straight lines. Briefly the algorithm proceeds by first classifying the boundary into line segments and curve segments and then finding the intersection points of these segments with horizontal scan lines. The segment width W is calculated from the x position of the intersection points. A histogram over the segment width provides the letter density distribution for that specific letter. Combining the histogram data and the probabilities associated with individual letters we obtain the font density distribution.

4.1 Converting Glyph Table into Segments

Our algorithm separates straight lines from Bezier curves using the property that two consecutive on points (marked by 1 in the Glyph table) indicate a straight line, whereas a curve is determined by the fact that an off point (marked by 0) is located between on points. The Glyph table is divided into two sets L and B, such that L = {L1, L2, ,LMl} and B = {B1, B2, ,BM}. L is a set of line segments where Ml the number of line segment and Lj is a line segment descriptor. Likewise, we note B = {B1, B2, ,BM}as a set of Bezier curve segments where Mb is the number of Bezier curve segment and Bk is a Bezier curve segment descriptor. The sets L and B may stored in matrices. L in a 4X Ml matrix and B in a 6X Mb matrix. The jth column of the matrix L(.,j) would store the jth line segment descriptor consisting of the (x,y) coordinates of two end points. Similarly the kth column of the matrix B(.,k) stores the kth Bezier curve segment descriptor of the boundary. The glyph table G is stored in a 3XM matrix where M is the total number of boundary points. Let the rows of this matrix e defined as : G_on(i), G(1,:) G_X(i), G(2,:), G_Y(i),G(3,:). We assume the table has been modified so that there are no off boundary points in sequence. Glyph Tables are often compressed by removing on boundary points between consecutive Bezier curve segments. The removed point may be deduced from two consecutive off boundary points by computing the mid points between them. The algorithm follows assumption that the Glyph table has been adjusted so that successive zeros do not occur in G_on(i).

4.2 Euler Number and Glyph Classification

We have seen that each letter consists of a glyph contour which may be classified into two categories such as simple and compound glyphs. Briefly speaking simple glyph letters can be drawn using only one boundary contour whereas complex glyph

letters require more than one boundary contour. In the alphabet, we can evaluate the Euler number of each letter by counting the number of differences between connected components and holes denoted by *El*

$$El = C - H \qquad 4\text{-}1$$

Where C is the number of connected components and H is the number of holes in the letter. By looking at individual capital letters, we find patterns, which determine their glyph classification as simple or compound glyphs. When $El = 1$ then the letter is a simple glyph letter and when $El > 1$ then the letter is a compound glyph letter.

4.3 Scan Line Width Matrix

The existence of intersection points is investigated when the height *h* falls into the range of *y* values of the line segment or the Bezier curve segment. Since we have already introduced the parametric equations for a straight line and Bezier curve equations, it is not difficult to calculate the t value for a given height *h* and segment table.

For straight lines :

$$t = \frac{h - A_y}{B_y - A_y} \qquad 4\text{-}2$$

Where $By-Ay \neq 0$ and $0 \leq t < 1$

For Bezier curves :

$$t = \frac{(A_y - B_y) \pm \sqrt{(B_y - A_y)^2 - (A_y - 2B_y + C_y)(A_y - h)}}{(A_y - 2B_y + C_y)} \qquad 4\text{-}3$$

Where $Ay-2By+Cy \neq 0$ and $0 \leq t < 1$. Using the valid *t* obtained above we can to calculate the *X* location of an intersection point. Each segment (line or curve) associated with the valid t keeps its *x* and *y* glyph data in *L(4, j)* and *B(6, k)* which provide *Ax, Bx* for a line segment and *Ax, Bx* and *Cx* for a curve segment.

$$X(c) = (1-t)^2 A_x + 2t(1-t) B_x + t^2 C_x$$
$$X(c) = (1-t) A_x + t B_x \qquad 4\text{-}4$$

The intersection points are maintained by a matrix *X(c)* where *X(c)* holds the *X* location of *cth* intersection point for a given height *h*. The final value of *c* is the total number of intersection points for a given height h denoted by *ch*. After sorting *X(c)*, the scan line width matrix is properly computed based on even and odd rules of intersection point index. Now we have calculated the *X* location of the intersection points for a height *h*, and stored those into *X(c)*. Our final goal is to build a histogram matrix over segment widths *W* so that we need to compute segment widths *W* from *X(c)*. A segment width is surrounded by outlines of fonts and defined by an interior distance as shown in Figure 1. Thus we have to recognize how to classify which intersection points are inside. The well known even and odd rules are adapted here [18]. The definition of *W* is as follows :

$$W = X(2c) - X(2c-1) \qquad 4\text{-}5$$

Where $1 = c = ch/2$ given a height h. Example 4.3 shows the details. In other words, we regards W as a distance between the x location of an even index intersection point and the x location of an odd index intersection point.

4.4 Histogram Matrix Over Scan line width

The histogram over the scan line width matrix is introduced in order to display the number of occurrences of various segment width W. This helps us to find which segment width appears with a high probability. This will allow us to make a judgements concerning the letter recognition system. By calculating and storing the scan line width into the proper bin, we are able to investigate its letter density for a given segment width and height. Table 1 shows TrueType features implying two dominant segment widths appear most frequently often for the Times Roman Capital. We note that higher number of occurrence of two dominant segment widths and their number of occurrence features are definitely employed as TrueType features for letter recognition followed by matching between TrueType feature library and input letter images.

4.5 Similarity Coefficient

By computing similarity coefficient of TrueType features, we are able to determined each letter associating with a font as comparing similarity coefficient. Similarity coefficient is defined by

$$c(l_r) = \frac{1}{|l_r - l_c| + 1} \qquad 4\text{-}6$$

Where lr is a set of reference feature namely TrueType feature library and lc is a candidate feature vector such as [5 28 2 24] for U (size 36) as shown in Table 1.

As a matter of fact perfect matching make similarity coefficient 1 since lr is equals to lc. Preprocessing steps should be required in order to separate letters given word images using projection followed by comparing similarity coefficient between TrueType feature library and input letters. Our simple edge detection followed by calculating histogram over segment width enable us to construct TrueType feature library. We have performed the a number of experiments in order to verify accuracy of the proposed features. In the size of 36, and 72, the proposed features are accurately matched whereas the small size of letters are less accurate. Figure 6 to Figure 9 demonstrates matching process to compare feature library(Times Roman Capital, size=12, 36, 72)with 'W','O','R',and 'D' respectively. In these Figures horizontal axis indicates 78 letters Times Roman 12, 36 and 72 size in sequence.

RECOGNITION

RECOGNITION

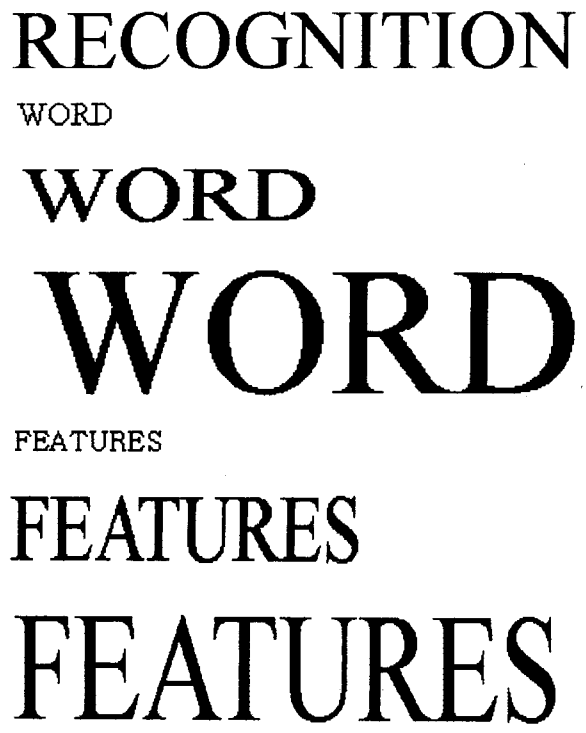

Figure 5. Sample Experimental Results 12, 36 and 72 size of Times Roman Capital

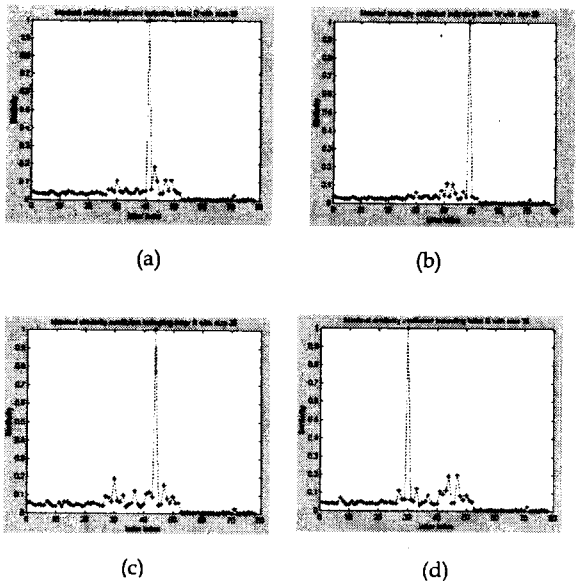

Figure 6. Maximal similarity coefficient 1 indicating letter 'W','O','R' and 'D' with size 36

In Figure 10 and 11, we demonstrate our proposed method is very suitable and novel of which sizes are 12, 36, and 72 respectively. In this experiments, we only perform to evaluate features for letter recognition Time Roman Capital. Some limitation have been found while doing experiments for small letters whose size is 12 due to digitization problem. For small letters, dominant segment widths are mostly 1 so that it is difficult to extracting accurately expected letters.

5. Conclusion and future work

We propose feature extraction for word recognition based on TrueType letter density function. Hence, we are deriving the segment with density for a letter with a specific TrueType, defined by the number of occurrence over a segment width. A certain number of occurrence appears frequently often due to fixed segment widths. By experiments, we utilize our approaches into letter recognition by comparing TrueType feature library of a letter with that from input word images. Experiments have been carried out to justify robustness of the proposed method showing acceptable results. This results in providing sufficient features for word recognition since each letter is almost uniquely identified by TrueType features. As future work, we still solve for small size letters and various fonts.

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

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

[4] "Microsoft Typography Quick Reference http://www.miscroft.com/typography/tools/tools.htm

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

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

[7] 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

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

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

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

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

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

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

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

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

[16] L. Penney, "A brief history of TrueType",A article on the True Type from interview with Kaasila, the Principal Inventor June,1997.

[17] D. Herman and the Apple True Type team, "The TrueType Reference Manual", Apple Computer Inc. Feb.1998

[18] A. Gross and L. Latecki, "Digital geometric method in document image analysis", The Journal of The Pattern Recognition 32 (1999) pp. 407-424, 1999

[19] yong Zhu, T. Tan and Y. Wang, "Font Recognition based om Global Texture Analysis", IEEE Transactions on Pattern Analysis And Machine Intelligence. Vol. 23. No. 10, pp. 1192-1200 October 2001

[20] Z. Yu and C. L. Tan,"Imaged-based Document Vectors for Text Retrieval", Int'l Conf. on pattern recognition, Barcelona, Spain, Sep. pp393-396, 2000

[21] Chen, F. R., Bloomberg, D. S "Extraction of indication Summary Sentence from imaged Documents", Proceeding 4th int'l conf. on Document Analysis and Recognition, ICDAR'97, Ulm, Germany, Vol. 1,pp 227-232, 1997

● 저자소개 ●

진성아

1991 전북대학교 자연과학대학 수학과 학사

1993 전북대학교 자연과학대학 전자계산학과 석사

1995 (미) City University of NewYork, Computer Science 대학원 수료

1999 (미) Stevens Institute of Technology, Computer Science 박사

1998~1999 (미) Stevens Institute of Tech. 강의조교

1998~1999 (미) NewJersey Institute of Tech. 연구조교

2000~2001 서강대학교 영상대학원 미디어공학과 연구교수

2001~현재 성결대학교 멀티미디어학부 전임강사

관심분야:

Computer Vision, Virtual Reality, Document Image Understanding.