International Journal of Mathematics and Statistics Invention (IJMSI)
E-ISSN: 2321 – 4767 P-ISSN: 2321 - 4759
www.ijmsi.org Volume 4 Issue 4 || April. 2016 || PP-06-12

Free-scale Magnification for Single-Pixel-Width Alphabetic Typeface Characters

Tianxiang Zheng
(Shenzhen Tourism College/Jinan University, China)

ABSTRACT ***:*** *This article presents a novel approach to magnify single-pixel-width alphabetic typeface characters. To this end, useless serif patterns of the character were first removed from the subsequent analysis to facilitate the zooming-in process and alleviate the computation burden. An intuitive algorithm for stroke transcribing was then advanced and applied to the serif-eliminated character. Finally, each stroke, which was represented by cubic B-Spline functions, was scaled by wavelet transform with arbitrary size. The proposed algorithm was validated on the computer fonts rendered with Roman, 12 point in Windows operation system. Experimental results demonstrate its good performance, which may provide convenient access to small devices and mobile interfaces for the future use.*

KEYWORDS - *Single-pixel-width Alphabetic Typeface, Serif Removal, Stroke Transcribing, Cubic B-Spline, Wavelet Transform*

I. INTRODUCTION

Advances in the power and availability of mobile technology, coupled with the "on-the-go" lifestyle of many individuals, have made small screen devices nearly ubiquitous in everyday life. For example, professionals regularly rely on mobile handheld devices (e.g., personal digital assistants (PDAs) or smart phones) to check emails, gather data, and make critical decisions while away from the home office. Non-professional uses include navigation, social networking, web surfing, and entertainment, again with the same goal of collecting information in task relevant and appropriate ways. One obvious tradeoff that occurs when reading textual information on small devices is the reduced character size utilized on these displays. While mobile devices enable convenient access to large amounts of information and data, literally at one's fingertips, is it possible for the user to enjoy enhanced personal flexibility, for example, to zoom in such a reduced-size word according to his/her interest to identify the characters?

The demand on small size displays behoves us to investigate scaling of the small size character on typeface. Among all kinds of character, typeface, constructed and output by computer fonts, is an important field of research and application. Over the years, many different techniques have appeared to computer fonts. So far, a large number of methods related to font processing have been reported in the literature in accordance with the different characteristics of different sizes and types[1-7]. Nevertheless, most of the progress made in typeface processing are dedicated to large size fonts, while the opposite is almost left untreated. As small size fonts find wider and wider applications, the need for developing special analyzing technique becomes more urgent.

Scaling is an example of the processing performed on the character input, including magnification and minification. It is well-recognized that the larger the font size is, the better the quality of scaling is. Unfortunately, scaling has been shown previously to negatively impact performance on these small-size fonts[8], which refers, specifically, to those sized at about 10 × 10 pixels between the 8 point and 12 point character size(see Fig. 4 for an example), likely because of little energy with narrow strokes(always single-pixel-width) and high sensitivity to noise. Moreover, magnification is generally considered to be more complicated than minification since errors made in one stage get compounded by the time the next stage is performed, leading to pronounced jagged edges. As a result, magnification on small-size fonts is a formidable problem in character processing.

This paper seeks to contribute to filling this gap by proposing a novel approach to perform magnification on small size fonts (typically, the single-pixel-width alphabetic typeface characters), so as, ultimately, to cater convenient access to small device holders. To this end, useless serif patterns of the character are first removed from the subsequent analysis to facilitate the zooming-in process and alleviate the computation burden. An intuitive and effective algorithm for stroke transcribing is then advanced and applied to the serif-eliminated character. Finally, each stroke, which is represented by cubic B-Spline functions, is scaled by wavelet transform with arbitrary size.

II. PRELIMINARIES

2.1 Serif

Serif, the specialized terminology in press and typesetting industry, refers to the short narrow stick orthogonal to the stroke added by the font designer for good-looking. Usually it occurs in some frequently-used fonts, such as Roman and Courier[8]. Figure 4 shows a good example for this, where three short narrow sticks were attached horizontally to the strokes at the bottom of the character.

2.2 Neighborhood

A neighbor that is a character point is named character neighbor, and a neighbor that is not a character point is named non-character neighbor[9]. There are two types of connectivity or neighborhoods in digital images: 4-connectivity or 8-connectivity. For the picture to be complete, we use 8-neighborhood from the remainder of this study.

2.3 Arm

An isolated character neighboring or a group of consecutive character neighbors of character point P is named arm of character point P [10].

2.4 Fork point

A fork point is a character point that has three or four arms[10].

2.5 Branch

A branch consists of a number of consecutive points on a certain stroke.

2.6 Trend

Let the consecutive points of a segmental branch be $P_1, P_2, \dots, P_k$. To avoid misleading results caused by the ill-conditioned effect while using a single method to measure the single-pixel-width character, the trend of the segmental branch used in this study is calculated as follows, which is fundamentally different from and more easy-to-manipulate than that of the previous work developed by Lin[10] where only $Trend^{[3]}$ below was adopted:

$$Trend = \sum_{j=1}^{3} w_j Trend^{[j]};$$

where $Trend^{[j]} = \tan^{-1}(Y^{[j]}/X^{[j]})$, $0 \le Trend^{[j]} < 2\pi$, $j = 1,2,3$ and

$$X^{[1]} = \sum_{i=1}^{\lfloor k/2 \rfloor} \frac{P_{(i+\lfloor k/2 \rfloor)x} - P_{ix}}{\left\| P_i P_{(i+\lfloor k/2 \rfloor)} \right\|}, \quad Y^{[1]} = \sum_{i=1}^{\lfloor k/2 \rfloor} \frac{P_{(i+\lfloor k/2 \rfloor)y} - P_{iy}}{\left\| P_i P_{(i+\lfloor k/2 \rfloor)} \right\|}$$

$$X^{[2]} = \sum_{i=1}^{\lfloor k/2 \rfloor} \frac{P_{(2i)x} - P_{(2i-1)x}}{\left\| P_i P_{(i+1)} \right\|}, \quad Y^{[2]} = \sum_{i=1}^{\lfloor k/2 \rfloor} \frac{P_{(2i)y} - P_{(2i-1)y}}{\left\| P_i P_{(i+1)} \right\|}$$

$$X^{[3]} = \sum_{i=1}^{\lfloor k/2 \rfloor} \frac{P_{(k-i+1)x} - P_{ix}}{\left\| P_i P_{(k-i+1)} \right\|}, \quad Y^{[3]} = \sum_{i=1}^{\lfloor k/2 \rfloor} \frac{P_{(k-i+1)y} - P_{iy}}{\left\| P_i P_{(k-i+1)} \right\|}$$

where (P_{ix}, P_{iy}) is the position of P_i and $\left\| P_i P_j \right\|$ is the length of vector (P_i, P_j) , w_j are weight values specified by user. In this study, we set $w_1 = 0.5$, $w_2 = 0.3$, $w_3 = 0.2$.

III. PROPOSED ALGORITHM

The essence of the proposed method stems mainly from the description of the single-pixel-width alphabetic typeface character. Once this is achieved, a highly efficient technique of zooming-in raised by Yang[11] could show its usefulness. Specifically, the following three steps constitute the proposed method:

(1) Remove the serif of the character

(2) Describe the character by a trend-followed stroke transcribing algorithm
(3) Magnify the character with arbitrary size

3.1. Serif Removal

Apparently, the troublesome serif is helpless in terms of zooming, so it is preferable to have it removed. By means of the method documented in [8], the serif can be eliminated from the subsequent analysis.

3.2. Trend-followed Stroke Transcribing

A character can be described by one of the modes of bitmap, vector outline, and contour. The contour approach has been shown to be superior to the others because of its simplicity and capability to be arbitrarily scaled, rotated, or stretched without sacrificing quality[1]. In this study, since it doesn't exist a real "contour" with regard to single-pixel-width character, we use the word "stroke" instead. That's why we choose contour-like here to give the description.

3.2.1. Pre-processing: Determine the location of fork points

Each character point is labeled the number of its arms to identify the fork point. To do this, the number of edges of the character neighbors is counted and then divided by two[10].

3.2.2. Find the starting point

To inspect the starting point, we adopt the following rule: first search for the point with the arm equal to 1. If fails, consider the neighborhood of those tracked points. If fails either, take the first point encountered from the scanline.

3.2.3. Transcribe the stroke forward

As mentioned above, the character is single-pixel-width and usually no close contour can be found, which appears to be more like a unicursal. A particular algorithm is therefore needed, of which the fork points located on the character is crucial. The major steps of the algorithm include:

(1) Label the fork point

(2) Transcribe each stroke from a starting point and record the tracks until finding an end point or the fork point that is justified to be the end of the stroke by calculating the trends

There are two issues about the transcription that still need to be addressed: (1) After the starting point, transcription is continued until a fork point is encountered. Based on two different strategies, a decision is made to determine whether to go through it or terminate the procedure on the current segmental branch, depending on whether a specified threshold for the variation of trend between the current and the candidate branch is satisfied or not(see Fig.1 for details). Only possible candidates can be added to the current branch, otherwise, it is part of another stroke and the transcription on this branch ought to be terminated. (2) To make a compensation for the fact that the starting point is not necessarily an end point(see the previous subsection), each stroke will be forwarded to further examination on whether it should be merged into other branches found before. If the starting or ending point on one branch locates nearby another branch, these two branches are actually an integrated one and should therefore be merged.

The transcription is repeated until all character points have been tracked. The results can be referred to in the next section, where the pattern is magnified without changing the number of strokes. The procedure explained above is summarized by the flowchart of Fig.1. More detail of the algorithm can be found in Fig.2 and Fig.3.

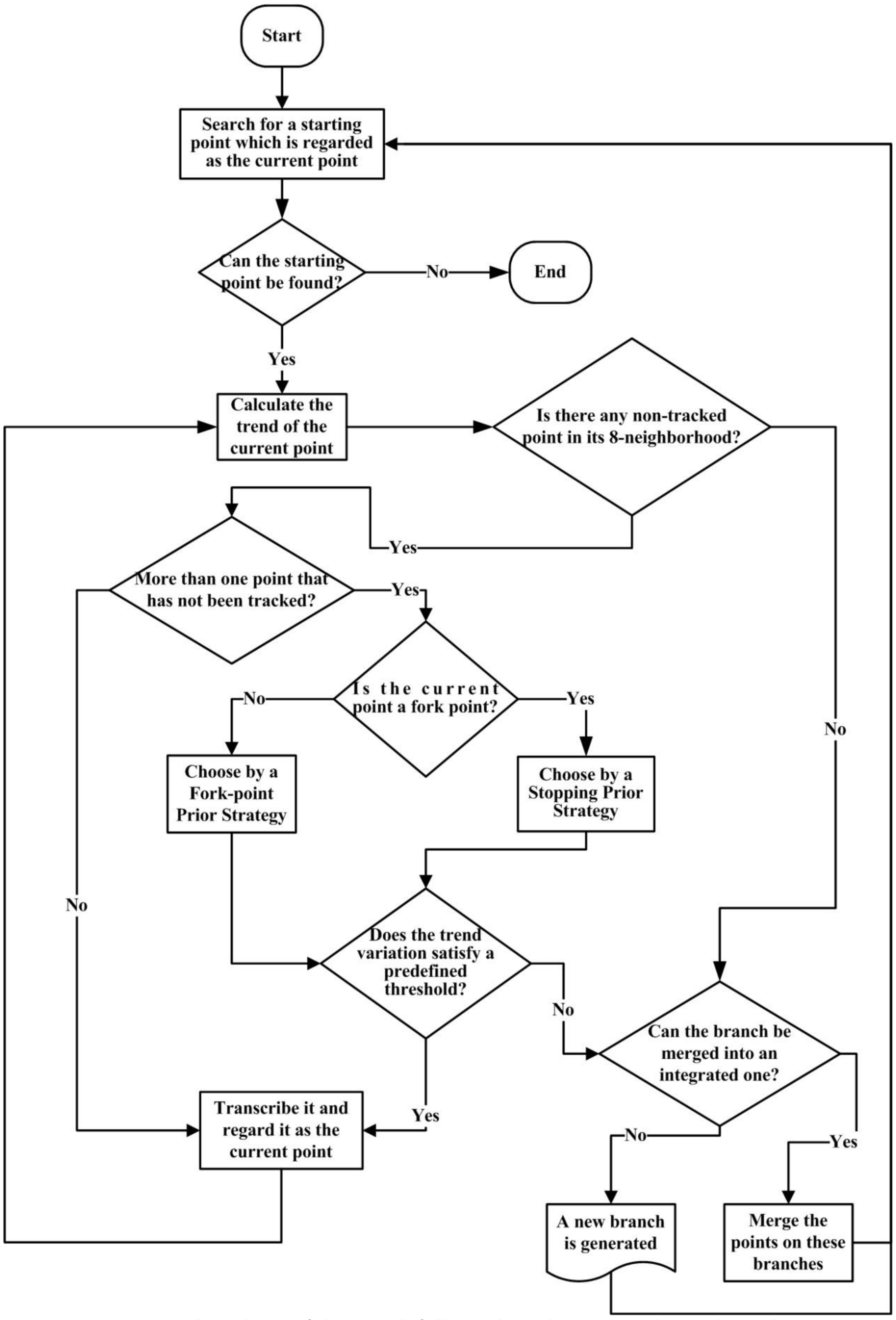
Start
Search for a starting point which is regarded as the current point
Can the starting point be found?
No
End
Yes
Calculate the trend of the current point
Is there any non-tracked point in its 8-neighborhood?
Yes
No
More than one point that has not been tracked?
Yes
No
Is the current point a fork point?
No
Yes
Choose by a Fork-point Prior Strategy
Choose by a Stopping Prior Strategy
Does the trend variation satisfy a predefined threshold?
No
Yes
Transcribe it and regard it as the current point
Can the branch be merged into an integrated one?
No
Yes
A new branch is generated
Merge the points on these branches

Figure 1 Flowchart of the trend-followed stroke transcribing algorithm

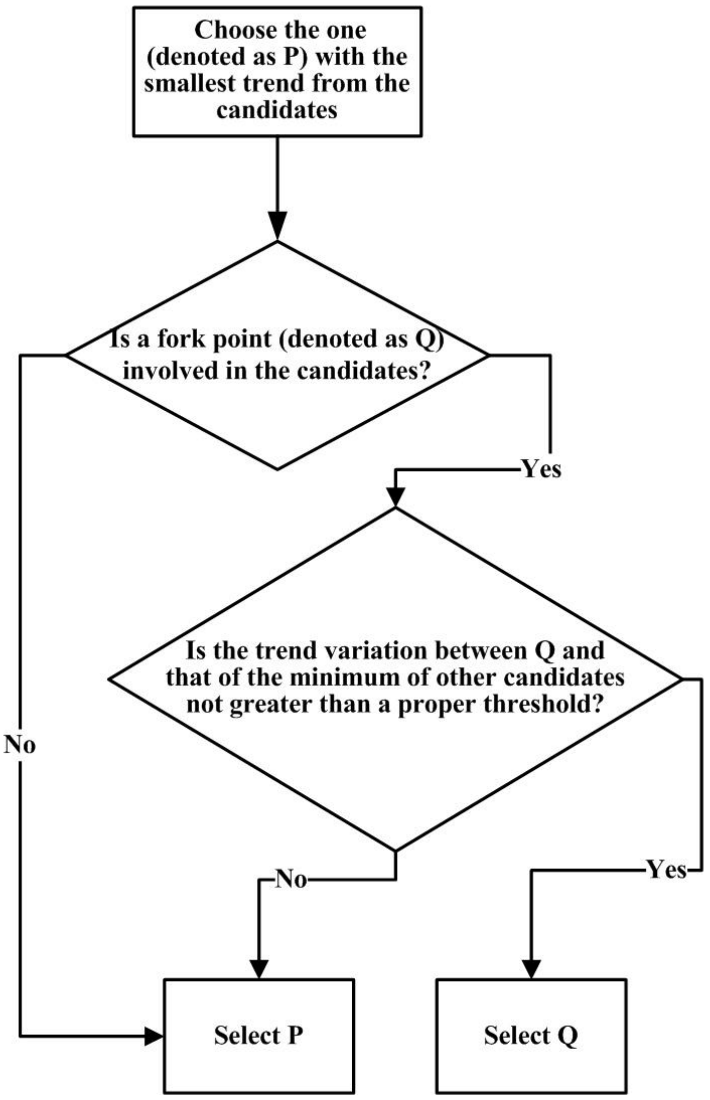
Choose the one (denoted as P) with the smallest trend from the candidates
Is a fork point (denoted as Q) involved in the candidates?
Yes
No
Is the trend variation between Q and that of the minimum of other candidates not greater than a proper threshold?
No
Yes
Select P
Select Q

Figure 2 Outspread of the module "Choose by a Fork-point Prior Strategy" in Fig.1

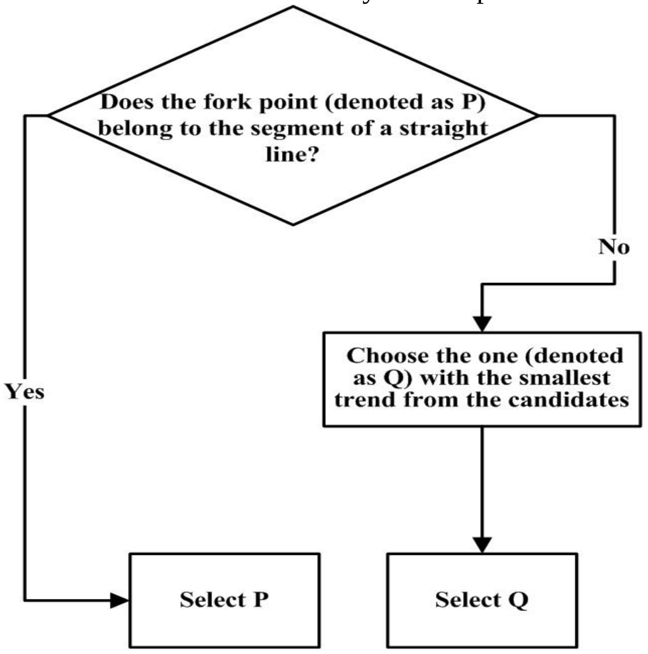
Does the fork point (denoted as P) belong to the segment of a straight line?
No
Choose the one (denoted as Q) with the smallest trend from the candidates
Yes
Select P
Select Q

Figure 3 Outspread of the module " Choose by a Stopping Prior Strategy" in Fig.1

3.3. Free-scale Zooming In

Yang built a B-Spline method for Chinese character[11], in which the contour was first characterized by cubic B-Spline functions and then scaled by a multi-resolution B-Spline wavelets. Such idea is introduced to the under-addressed problem in the case of stroke generated above.

IV. EXPEIRIMENTAL RESULTS

The text samples being tested come from rendering directly in Windows platform, as illustrated in Fig.4. Obviously, the appearance of the jaggies is significant in single-pixel-width character that can not be neglected during scaling, since it becomes conspicuous after the character is magnified.

From the experimental results, we can see that the proposed method for trend-followed stroke transcribing is capable of accurately diagnosing the contour-like strokes of the single-pixel-width character. Due to the use of the filter procedure of wavelet transform, the distortion of the enlarged type is very light and it correlates only to the precision of the sample of the original image itself. After scaling, the shape and the essential structure are well maintained, neither inducing any additional distortion nor sacrificing understanding of the character.

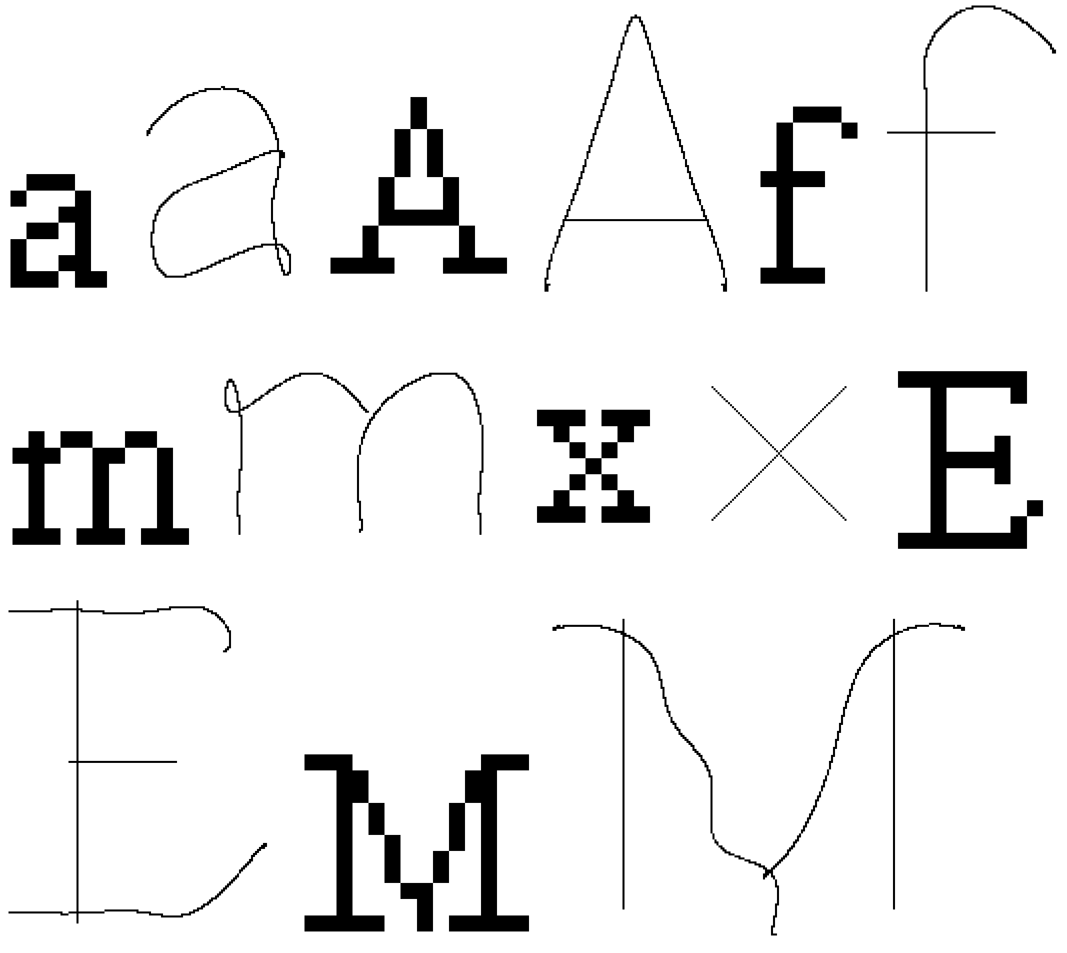

Figure 4. Emulated characters and results by using the proposed algorithm. For each group, the left image is the original character rendered with the Roman font, 12 point in Windows operation system. To clarify the pixels, all the raw characters were displayed by an 8-times scaling with the tool "mspaint" in Windows, namely, one pixel corresponds to one "block" area. The right image is the magnification result of the left one at 15 times with the proposed method. To improve the quality of the enlarged type, the wavelet filter was utilized, where the initial layer was set to 5(see [11] for more details).

V. CONCLUSION

The small screen devices and mobile interfaces have provided new grounds for small-size character scaling which is an important issue in typesetting and graphical text. Since the literature shows that few efforts have been made to contribute to the problem, it should be meaningful to approach the task of examining the magnification on small-size characters. This study advances a novel approach for free-scale magnification on single-pixel-width alphabetic typeface characters, which allows users to tailor the devices for modifying the displayed text size as needed. To do this, an intuitive trend-followed stroke transcribing algorithm, analogous in conception to contour tracing, is pioneered. To facilitate the differentiation between characters when the character size is small to remain more readable for users, this magnification is practically amenable.

A perhaps less-attractive application on the finding of trend followed stroke transcribing in this study would be in the area of skeleton on thinning, which up to now only few essential structures, such as fork points, corner points and other scattered points, have been employed to the procedure of feature extraction. Full usage of the skeleton, if possible, will definitely enhance the subsequence classification and recognition. By means of the transcribing algorithm raised by the author, the sufficient use of the entire skeleton that allows researchers or practitioners to achieve better results on character segmentation or recognition previously not possible via a certain scattered points can therefore be expected.

Last but not least, although our goal is dedicated to alphabetic typeface, it is applicable to single-pixel-width numerical typeface, or even handwritten characters since there is no inherent difference between them, which in turn results in no limitation on applying the proposed algorithm.

ACKNOWLEDGEMENTS

This work was partially supported by the 17th Special Funds of Undergraduate Course Center of Educational Reform Research Project of Jinan University in 2015 under Grant No.JG2015094.

REFERENCES

[1] A. Namane and M. A. Sid-Ahmed, Character scaling by contour method. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12(6), 1990, 600-606.

[2] W. K. Pratt, *Digital Image Processing* (New York: Wiley Interscience, 1978).

[3] A. Shamir and A. Rappoport, Feature-Based Design of Fonts Using Constraints. *Lecture Notes in Computer Science,* 1375, 1998, 93-108.

[4] B. Stamm, Visual truetype: A graphical method for authoring font intelligence. *Lecture Notes in Computer Science,* 1375, 1998, 77-92.

[5] R. A. Ulichney and D. E. Troxel, Scaling Binary Images with the Telescoping Template. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 4(3), 1982, 331-335.

[6] U. Schneider, An object-oriented model for the hierarchical composition of letterforms in computer-aided typeface design. *Lecture Notes in Computer Science,* 1375, 1998, 109-125.

[7] P. Karow, *Font Technology: Description and Tools* (New York: Springer-Verlag, 1994).

[8] J. X. Guo and L. Yang, A Novel Approach to Segment Multi-size Machine Printed Characters by Removing Serifs. *Pattern Recognition And Artificial Intelligence,* 19(6), 2006, 702-707.

[9] M. Cheriet, N. Kharma, C. L. Liu, and C. Y. Suen, *Character recognition systems: a guide for students and practitioners* (New York: Wiley Interscience, 2007).

[10] J. R. Lin and C. F. Chen, Stroke extraction for chinese characters using a trend-followed transcribing technique. *Pattern Recognition,* 29(11), 1996, 1789-1805.

[11] Y. Yang, P. Xiao, and Y. L. Yu, A free-scaling algorithm for chinese fonts based on cubic B-Spline wavelet transform. *Chinese Journal of Computers,* 21(12), 1998, 1141-1445.