Text vectorization based on character recognition and character stroke modeling

Fan, Zhigang, Zhou, Bingfeng, Tse, Francis, Mu, Yadong, He, Tao

Zhigang Fan, Bingfeng Zhou, Francis Tse, Yadong Mu, Tao He, "Text vectorization based on character recognition and character stroke modeling," Proc. SPIE 9027, Imaging and Multimedia Analytics in a Web and Mobile World 2014, 902707 (3 March 2014); doi: 10.1117/12.2045596

SPIE.

Event: IS&T/SPIE Electronic Imaging, 2014, San Francisco, California, United States

Text Vectorization Based on Character Recognition and Character Stroke Modeling

Zhigang Fan[1], Bingfeng Zhou[2], Francis Tse[1], Yadong Mu[2], Tao He[2],

[1]Xerox Corp., Webster, NY, USA
[2]Peking University, Beijing, China

ABSTRACT

In this paper, a text vectorization method is proposed using OCR (Optical Character Recognition) and character stroke modeling. This is based on the observation that for a particular character, its font glyphs may have different shapes, but often share same stroke structures. Like many other methods, the proposed algorithm contains two procedures, dominant point determination and data fitting. The first one partitions the outlines into segments and second one fits a curve to each segment. In the proposed method, the dominant points are classified as "major" (specifying stroke structures) and "minor" (specifying serif shapes). A set of rules (parameters) are determined offline specifying for each character the number of major and minor dominant points and for each dominant point the detection and fitting parameters (projection directions, boundary conditions and smoothness). For minor points, multiple sets of parameters could be used for different fonts. During operation, OCR is performed and the parameters associated with the recognized character are selected. Both major and minor dominant points are detected as a maximization process as specified by the parameter set. For minor points, an additional step could be performed to test the competing hypothesis and detect degenerated cases.

Keywords: text vectorization, outline fonts

1. INTRODUCTION

The shape of a text character can usually be represented in bitmap or outline (vector) forms. In the latter representation, a character is specified with a set of curves describing its outlines. Outline fonts exist extensively in electronically created files. However, they are not native for documents captured by digital cameras or scanners.

Vectorization of text is a process that converts the bitmaps of characters to their vector representations. Specifically, it uses a set of curves to reconstruct the characters. Compared to its bitmap, the vectorized text has the advantages of: 1) Resolution independency: the text can be easily scaled to different output resolutions. This is vital particularly for mobile devices. 2) Better image quality: the vectorized text appears smooth while bitmaps looks often jagged and bumpy. 3) Easy editability: Shape of the text can be edited using standard graphic tools. This enables easy modification of font attributes (size, boldness, etc) for repurposing.

The research of curve vectorization and the related topic of dominant point detection have a very long history. The early works include [1-3]. Numerous algorithms have been proposed since then [4-19]. They can be roughly classified into two groups, high curvature point detection and polygonol approximation. Almost all of the proposed methods are designed to deal with vectorization of general curves, not particularly for text characters. They usually do not perform well on text, as text vectorization often needs reproducing minute details (e.g. serifs) which are comparable to noise in size. Most existing text vectorization products in the market tend to over-fit the data and produce jagged outlines. The artifacts may not be visually unpleasant in many MFD (Multifunction Device) document copying/scanning applications, in which characters are often observed at their original sizes. However,

Imaging and Multimedia Analytics in a Web and Mobile World 2014, edited by Qian Lin, Jan P. Allebach, Zhigang Fan, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 9027, 902707 · © 2014 SPIE-IS&T · CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2045596

they could become more disturbing in mobile application, in which the text is more often reduced and enlarged via pinch and stretch gestures.

Text vectorization is typically composed of two procedures, dominant point determination and data fitting. The outline(s) of a vectorized character is (are) usually partitioned into more than one curve, and the terminal points (starting and ending points) of the curve(s) are specified by the dominant points. The outline between neighboring dominant points is fit by one curve (typically 1st – 3nd order polynomials). The dominant point determination and data fitting could be performed separately, or combined in an iterative manner. First, a set of dominant points, which usually represent corners and sharp transitions in the outlines of the characters are located. The segments between neighboring dominant points are then coded by one or multiple curves using data fitting. Each segment is first fitted with one curve. If the fitting error exceeds expectation, additional dominant points are introduced to partition the segment into smaller segments, and each of them is fitted by one curve. Further partition may occur if the fitting error is still off target.

The number and the positions of the dominant points are essential for the quality of vectorization, as this is demonstrated in Figure 1 with a simple example with a san serif font "I". Figure 1a) gives the ideal detection, with four dominant points (represented by red dots) at the four corners of the vertical stroke. Figure 1b) and 1c) illustrate two kinds of errors, under-fitting and over-fitting. In Figure 1b), one dominant point is missing. As the outline between the dominant points 1 and 2 contains a corner and is very difficult to be fitted by one smooth curve, large fitting error can be expected. On the hand, over-fitting can also occur. An example is shown in Figure 1c). The three dominant points at the middle of the stroke fit the noise.

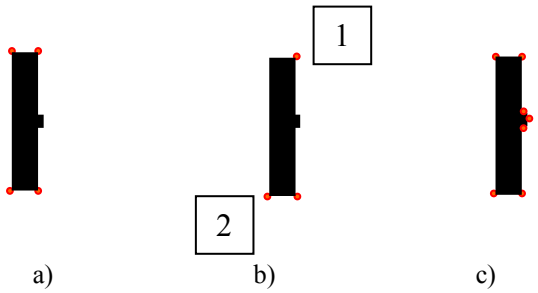


Figure 1. a) ideal dominant points; 2) under-fitting: missing one dominant point; 3) over-fitting

The dominant points are traditionally selected at the high curvature points. To avoid over-fitting the noise, smoothing filtering is often applied. The selection of the filter support size needs to meet two conflicting requirements. It should be large enough to smooth out the noise, and yet small enough so that two closely located high curvature points are distinguishable. The method generally works reasonably well for graphical objects when the distances between the neighboring dominant points are much larger than the noise size. However, it is difficult to produce satisfactory results for text, particularly the text with normal or small sizes. One reason is the text usually contains small details (particularly in serifs) that are comparable to noise in dimension. Another reason is many dominant points in text are not of high curvature in nature, for example, the points represented by blue dots in Figure 2. They reside in smoothly varying outlines, but are essential for a decent vectorization. They partition the inner and outer outlines, each into two curves. Without them, each circular outline needs to be fit with one curve, which is sometimes difficult.

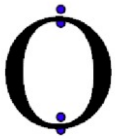


Figure 2. Dominant points on smooth curves.

It would certainly be preferable that a smooth curve in the original remains smooth in its vector representation. However, this is often not the case in the traditional methods. As the adjacent segments are reconstructed independently from each other, there is no guarantee if they will be connected with a smooth or sharp transition. Artificial jaggedness or discontinuities could be introduced at the boundary between the neighboring segments as a result. This is illustrated in Figure 3.

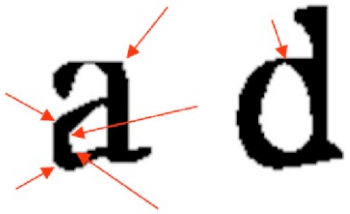


Figure 3. Artificial discontinuities (pointed by the red arrows) introduced in conventional text vectorization.

In this paper, a text vectorization method is proposed using OCR (Optical Character Recognition) and character stroke modeling. This is based on the observation that for a particular character, its font glyphs may have different shapes, but often share same stroke structures. In the proposed algorithm, a set of rules and parameters are determined offline for each character. During operation, OCR is performed and the rules/parameters associated with the recognized character are selected and used both for dominant point determination and data fitting. Note that in mobile applications, the offline determination can be done in the cloud to reduce computation burden on the local handheld device while employing more capable OCR engine to assist with the text vectorization.

The rest of the paper is organized as follows. Sections 2 briefly introduces the proposed method, with Sections 2.1 and 2.2 providing more details about the dominant point detection and data fitting, respectively. Section 3 describes the experimental results.

2. PROPOSED ALGORITHM

In this paper we propose to use OCR (Optical Character Recognition) results to guide the process of vectorization. OCR is often performed for many reasons (making document searchable, metadata extraction, form recognition, categorization, etc.). Consequently, using OCR for text vectorization may not necessarily introduce extra computation costs.

In the proposed method, the dominant points are classified as “major” (specifying stroke structures) and “minor” (specifying serif shapes). An example of major and minor dominant points are shown in Figure 4a) and b). A set of rules and parameters are designed offline determining for each character the number of major and minor dominant points and for each dominant point the detection parameters (projection directions and boundary conditions). For minor points, multiple sets of parameters could be used for different fonts. Both major and minor dominant points are further specified as *sharp* or *smooth*, indicating if a sharp change in curve direction may occur at the dominant point. During the operation, OCR is performed and the rules/parameters associated with the recognized character are selected. Both major and minor dominant points are detected as a maximization process as specified by the parameter set. For minor points, an additional step may be performed to test the competing hypothesis and detect degenerated cases. The sharpness properties of the dominant points are utilized in data fitting to further ensure reconstruction fidelity.

a) b)

Figure 4. Examples of major and minor dominant points: a) major dominant points (red dots); b) minor dominant points (blue dots). (enlarged, only the bottom left corner of the character is shown).

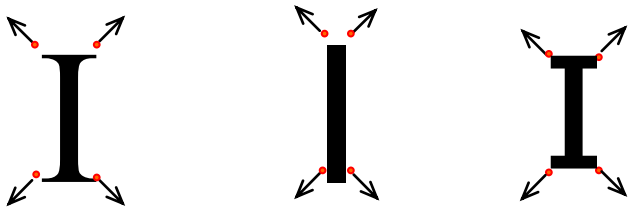
The algorithm is based on the following observations:

1. For a particular character, its font glyphs may have different shapes, but they often share same stroke structures.
2. The dominant points can be classified into two types: a) the "major" ones that define starting, ending, and intersection points of the strokes; b) the "minor" ones that specify the shapes of the strokes (e.g. serifs).
3. As a result of 1 and 2, the major dominant points are relatively stable (independent of fonts) and their detection can be significantly improved by exploiting OCR results.
4. The dominant points are usually maximum and minimum points when the boundary is projected to a certain direction. The maximum and minimum can be either global or local.

2.1 Dominant point detection

The proposed algorithm for dominant point detection is best explained with a simple example. Figure 5 shows letter "I" with three different fonts. Although they appear differently, their stroke structures are identical: a vertical stroke. Four major dominant points can be obtained as the Northwest, Northeast, Southwest, and Southeast corners (the red dots) of the glyphs. Specifically, they can be calculated as

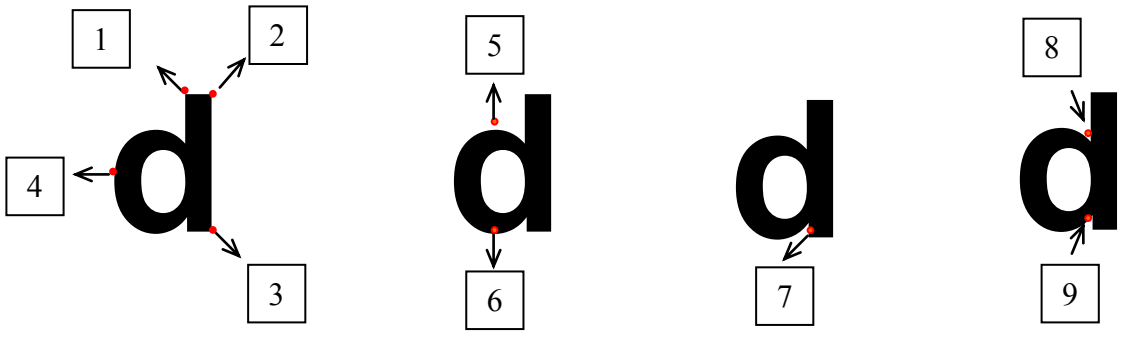
$$arg\ max_{(x,y)}\ x\ cost + y\ sint \qquad (1)$$



(a) (b) (c)

Figure 5. Major dominant point detection

where search is performed on all outline points (x,y) of the glyph, t is a parameter that determines the projection direction, along which we attempt to maximize. In this example, t is $3\pi/4$ (for NW corner), $\pi/4$ (NE), $5\pi/4$ (SW), and $7\pi/4$ (SE) respectively (the arrows in Figure 5).



(a) (b) (c) (d)

Figure 6. More complex example for major dominant point detection

Figure 6 shows a more complex example with letter "d". The major points are detected in several steps. In the first step, four points (labeled as 1-4) are determined using formula (1) with parameter $t = 3\pi/4$, $\pi/4$,$7\pi/4$, and π, respectively. (Figure 6a). In the second step (Figure 6b), dominant points 5 and 6 are detected with $t = \pi/2$ and $3\pi/2$, respectively. An additional restriction is imposed in maximum searching, which requires $x < W/2$ and $y > 0.8\ H$, where W and H are the width and height of the character, respectively. Point 7 is obtained in the third step (Figure 6c), in which $t = 5\pi/4$. There is also additional constraints. It is a local maximum "between" points 6 and 3. The relationship "between" is defined as follows. If a boundary loop is chain-coded (say in clockwise direction) and each boundary point is assigned an index according to the coding sequence. Point a is "between" points b and c if and only if:

$$I_c \geq I_a \geq I_b \text{ or } I_a \geq I_b \geq I_c \text{ or } I_b \geq I_c \geq I_a \qquad (2)$$

where I_a, I_b, and I_c are the indices of points a, b, and c, respectively. In the last step, two dominant points are determined. Unlike the previous points, points 8 and 9 reside at the concave parts of the boundary. As a result, the maximization in (1) should be replaced by minimization. Point 8 is between points 5 and 1, and point 9 is between points 7 and 6 (Figure 6d).

The dominant points for the inner loop can be easily located with $t = \pi/3$, $\pi/2$, $3\pi/2$ and $5\pi/3$.

For some letters, more than one set of rules are required for dealing with different glyphs or font variations. In those cases, the character is fitted with several sets of rules and the one that best fits the data is selected as the final result. Additional rules are also needed for italic letters.

Once the major dominant points are determined, the minor dominant points can be located in a similar manner. For example, the left two minor points of Time-Roman "I" in Figure 7a) (the minor points are represented by blue dots) can be determined by searching using (1) between the two left major points, with $t = 13\pi/12$ and $11\pi/12$, respectively. This will find the points where the serifs join the vertical stroke. There is a difference between the minor and major points. The minor points usually have more variations among different fonts. There are two consequences. First, we may need more models for one character. Specifically, we will determine the minor points using multiple sets of parameters and select the one that provides the best data fitting. Second, we may encounter many "degeneration" cases. This happens particularly if we want to share a same model for both serif and san serif fonts. This is illustrated by Figure 7b). This san serif font "I" does not contain any minor critical points. If the same rule used for serif font is applied ($t = 13\pi/12$ and $11\pi/12$), two artificial "minor points" will be detected at the same location as the major critical points (or in their vicinity). Typically, the detected points in the degeneration can be easily determined and deleted. Even if they are not discarded, these degenerated minor points do not cause noticeable image quality damages.

(a) (b)

Figure 7. a) Minor dominant point detection; 2) degeneration case.

OCR may occasionally produce errors. In that case, the dominant points detected may not be correct either. This will result in poor data fitting. A more than usual data fitting error will indicate something is wrong. In that case, the

system may use a default (say conventional) vectorization method. More importantly, by testing the quality of the data fitting, a false substitution of a wrong character can be prevented.

2.2 Data Fitting

In the proposed algorithm, a sharpness property is also determined from the glyph typography for each dominant point. The dominant point is *sharp* if a sudden change of curve direction may occur at the point. It is *smooth* if a smooth transition is expected.

During the process of vectorization, the dominant points are first detected, which partition the character outlines into segments. Each segment is then vectorized, depending on the smoothness properties of the dominant points at the both ends. If both dominant points are *sharp*, the segment is vectorized, independent of other segments. A curve is estimated that best fit the data in the segment. If at least one of the dominant points is *smooth*, boundary smoothness conditions are imposed. The actual implementation depends on the curve family used for coding.

A commonly used curve is the cubic Bezier curve:

$$Bx(\theta) = (1-\theta)^3 x_0 + 3\theta(1-\theta)^2 x_1 + 3\theta^2(1-\theta) x_2 + \theta^3 x_3$$
$$By(\theta) = (1-\theta)^3 y_0 + 3\theta(1-\theta)^2 y_1 + 3\theta^2(1-\theta) y_2 + \theta^3 y_3 \qquad \theta \in [0,1] \qquad (3)$$

where x_0, x_1, x_2, x_3, y_0, y_1, y_2, and y_3 are parameters that need to be estimated.

For cubic Bezier curve, if the dominant points at both ends are *smooth*, we have the flowing constraints:

$$x_0 = u_0, \; y_0 = v_0, \text{ and } (y_1 - y_0) / (x_1 - x_0) = d_0 \qquad (4)$$
$$x_3 = u_1, \; y_3 = v_1, \text{ and } (y_2 - y_3) / (x_2 - x_3) = d_1 \qquad (5)$$

where (u_0, v_0) and (u_1, v_1) are the coordinates of the boundary points and d_0 and d_1 are their derivatives. If one of the ends is *smooth*, while the other end is *sharp*, only one of (4) and (5) is applied. Figure 8 compares the results obtained with (Fig. 8b) and without smooth constraints (Fig 8a).

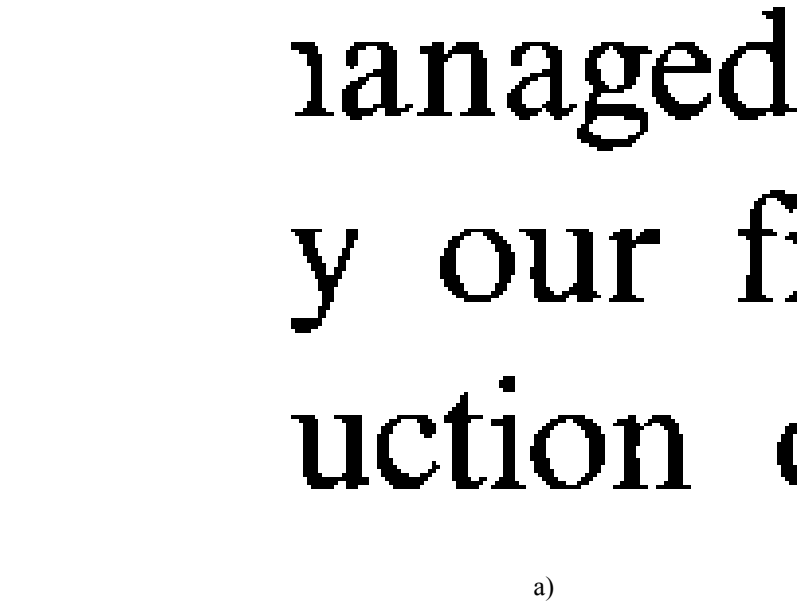
a) b)

Figure 8. Data fitting: a) without smoothness constraints; b) with smoothness constraints.

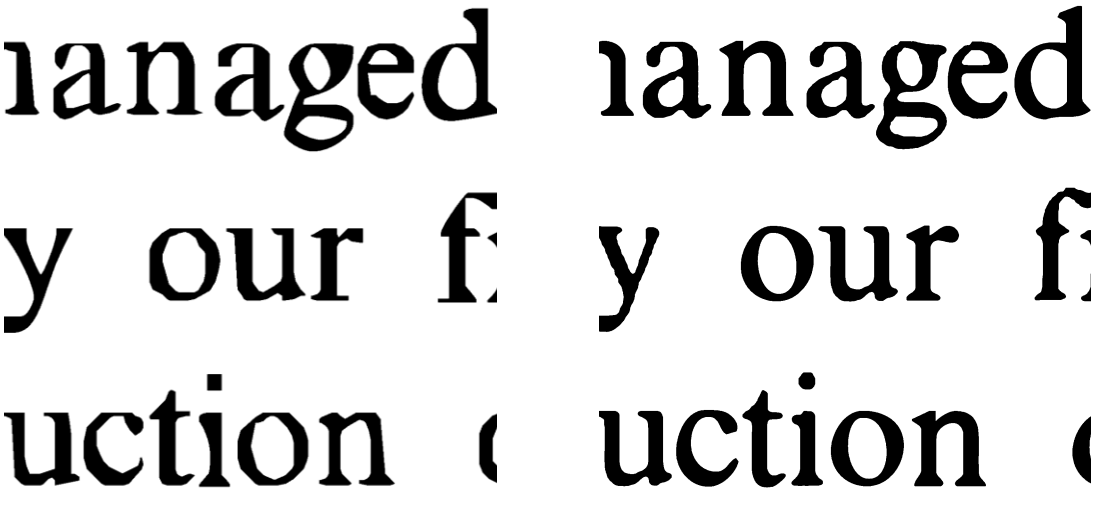
3. EXPERIMENTAL EXAMPLES

An experimental example is shown in Figure 9a)-c). Figure 9a) is the input bitmap. Figure 9b) shows the vectorization result generated by a commercially available product. Artifacts caused both by over-fitting and under-fitting can be observed. In additional to the jagged outlines, a few font details have been smoothed out (e.g. the top

left part of letter “f” and the top left part of the vertical stroke of letter “i"). Figure 9c) shows the result produced by the proposed method.



a)



b) c)

Figure 9. a) input bitmap; b) vectorization result (commercial product); c) vectorization result (proposed method)

REFERENCES

[1] E. Attneave, Some informational aspects of visual perception, Psychol. Rev. 61 (3) (1954) 183-193.
[2] A. Rosenfeld, E. Johnston, Angle detection on digital curves, IEEE Trans. Comput. 22 (1973) 940-941.
[3] Teh and R. Chin, "On the Detection of Dominant Points on Digital Curves", IEEE, PAMI, 11. 7, pp 859-872, 1989.
[4] B. Kerautret, J.-O. Lachaud, B. Naegel, Comparison of discrete curvature estimators and application to corner detection, in: ISVC (1), Vol. 5358 of LNCS, 2008, pp. 710-719.
[5] M. Marji, P. Siy, A new algorithm for dominant points detection and polygonization of digital curves, Pattern Recognition 36 (10) (2003) 2239-2251.
[6] D. Sarkar, A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves, Pattern Recognition Letters 14 (12) (1993) 959-964.
[7] T. M. Cronin, A boundary concavity code to support dominant point detection, Pattern Recognition Letters 20 (6) (1999) 617-634.
[8] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Trans. Electron. Comput. 10 (1961) 260-268.
[9] B. K. Ray, K. S. Ray, Determination of optimal polygon from digital curve using L1 norm, Pattern Recognition 26 (4) (1993) 505-509.
[10] A. Kolesnikov, P. Franti, Polygonal approximation of closed discrete curves, Pattern Recognition 40 (4) (2007) 1282-1293.
[11] H. Aoyama, M. Kawagoe, A piecewise linear approximation method preserving visual feature points of original figures, CVGIP: Graph. Models Image Process. 53 (5) (1991) 435-446.
[12] A. Held, K. Abe, C. Arcelli, Towards a hierarchical contour description via dominant point detection, Systems, Man and Cybernetics, IEEE Transactions on 24 (6) (1994) 942-949.
[13] M. Sarfraz, M. Asim, A. Masood, Piecewise polygonal approximation of digital curves, Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on (2004) 991-996.
[14] S.-C. Huang, Y.-N. Sun, Polygonal approximation using genetic algorithms, Pattern Recognition 32 (8) (1999) 1409-1420.
[15] H. Zhang, J. Guo, Optimal polygonal approximation of digital planar curves using meta heuristics, Pattern Recognition 34 (7) (2001) 1429-1436.
[16] P.-Y. Yin, Ant colony search algorithms for optimal polygonal approximation of plane curves, Pattern Recognition 36 (8) (2003) 1783-1797.
[17] L. Li, W. Chen, Corner detection and interpretation on planar curves using fuzzy reasoning, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (11) (1999) 1204-1210.
[18] A. Masood, Dominant point detection by reverse polygonization of digital curves, Image Vision Comput. 26 (5) (2008) 702-715.
[19] T. P. Nguyenc, I. Debled-Rennesson, A discrete geometry approach for dominant point detection, Pattern Recognition 44 (1) (2011) 32-44.