Using Area Voronoi Tessellation to Segment Characters Connected to Graphics

Yalin Wang† Ihsin T. Phillips‡ Robert Haralick†

†Dept. of Elect. Eng. Univ. of Washington, Seattle, WA 98195 U.S.A.

‡Dept. of Computer Science Queens College, CUNY Flushing, NY 11367 U.S.A.
{ylwang, yun, haralick@isl.wtc.washington.edu}

Abstract - Little attention has been paid to character connection problems in interpreting the engineering drawings. In this paper, three types of character connection problems are discussed and a method using area Voronoi tessellation is proposed to solve problem type II. Using area Voronoi tessellation, we can efficiently determine the contour of the characters and then detect the existence of any characters connected to graphics by hypothesis and validation. Projection analysis is used to segment and validate the characters connected to graphics. The correctness and feasibility of this method is demonstrated.

I. Introduction

Many algorithms were addressed to segment and recognize the character strings in interpreting line images such as maps and engineering drawings [1]-[6]. In [1], M. Burge and G. Monagan wrote "Nakamara et al. [2] give five reasons why character string extraction is difficult in topographic maps: characters often touch background figures, existence of many character like figures, various orientation of strings, intra-character spacing is different from string to string, and character strings are often close together." Among all the difficulties, the first problem was always ignored by the researchers. The reasons are in two-fold. First, it is difficult to detect and segment them with graphics. Secondly, their sparse occurrences do not hurt the recognition accuracy too much. Of course, the first reason is the major one.

There are still some studies concerned with this topic. R. Casey et al. [3] proposed an algorithm which was specifically designed for the intelligent form processing and could segment the characters connected to the form lines. Kasturi et al. also showed us an algorithm to recognize text connected to graphics in [4]. By growing up a three-sided box around the free sides of the component which was between two characters, the algorithm finally detected the character connected to the underline. Clearly, it was designed for a special case and it would be inefficient and time-consuming if used generally.

We believe the paramount problem to segment the characters connected to graphics is to locate an initial position to start the search for such characters. In [6], a system that segments and recognizes the character strings in the assembling drawings was implemented. The recognition accuracy rate was computed by comparing each recognized character with ground truth data and the desired recognition accuracy was no less than 98%. We propose and demonstrate a new method using the area Voronoi tessellation to detect the characters connected to graphics.

In this paper, we present the definition of area Voronoi tessellation in Section II. In Section III, we discuss an algorithm using the area Voronoi tessellation to segment the characters connected to graphics. Finally, an analysis of our algorithm time complexity serves as the conclusion in Section IV.

II. Area Voronoi Tessellation

The concept of Voronoi diagram is more than a century old, discussed in 1850 by Dirichlet and in a 1908 paper of Voronoi. In a sense, a Voronoi diagram records everything that one would ever want to know about proximity to a set of points or more general objects. Often one does want to know the detail about proximity: who is closest to whom, who is furthest, and so on. Voronoi diagram can help us. The dual graph of Voronoi Diagram is Delaunay triangulation.

Def. Let $P=\{p_1, p_2, \ldots, p_n\}$ be a set of points in the two -dimensional Euclidean plane. *Point Voronoi Diagram* $V(p_i)$ consists of all the points at least as close to p_i as to any other sites: $V(p_i)=\{x:|p_i-x|\le|p_j-x|, \forall j\ne i\}$.

To extract characters from engineering drawings, we should use a generalization of point Voronoi diagram: area Voronoi tess ellation. We use the definition of the area Voronoi tessellation presented by O. Boots, and Sughihara [5], which is as follows.

Def. Given that $A_1,\ldots,A_n$ are image elements and that p and q are locations in the image, we can define the distance, $d_a(p,A_i)$, from p to A_i as:

$$d_a(p,A_i)=\min_{q\in A_i} d(p,q)$$

This represents the minimum Euclidean distance from p to any location in A_i. Using this d_a, the area Voronoi region, $V_a(A_i)$, is defined as the set of locations from which the distance to A_i is less than or equal to the distance to any other areas:

$$V_a(A_i)=\{p|d_a(p,A_i)\le d_a(p,A_j), j\ne i, j=1,\ldots,n\}$$

For brevity, we will let $N_i=V_a(A_i)$, and the *area Voronoi tessellation* is the set $\gamma=\{N_1,\ldots N_i\}$. Fig. 3 shows the approximated area Voronoi tessellation of one circle and an oval.

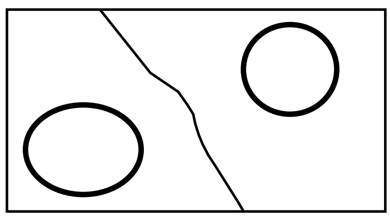


Fig. 3 Approximated area Voronoi tessellation of two sampled circles

The implementation of area Voronoi tessellation was given by M. Burge and G. Monagan in [6].

1. Image element sampling

Given a segment $\overline{P_1P_2}$ with two distinct endpoints $P_1=(X_1,Y_1)$ and $P_2=(X_2,Y_2)$, compute a sample point $P_3=(X_3,Y_3)$ such that for some α in the range $0\le\alpha\le1$, and some randomly selected perturbation γ, which is designed to avoid four cocircular sites, we have:

$$X_3=(\alpha X_1+(1-\alpha)X_2)\gamma$$

$$Y_3=(\alpha Y_1+(1-\alpha)Y_2)\gamma$$

where for a sampling rate S, α ranges from 0 to 1 by intervals of $1/S$. R is a uniformly distributed random number in the range $-0.5\le R\le0.5$ scaled by some factor D with $\gamma=RD$. D is dependent on the resolution at which the image was scanned. The sampled points are assigned the label of the component whence they came, $C(p)=$ component label.

2. Methods to get the Delaunay triangulation

Both Divide and Conquer method [8] and Fortune's algorithm [9] are available to us. The worst -case complexity of them is $O(n\log n)$.

3. Removable Delaunay quads

The Delaunay triangulation of the points must be processed to union two adjacent Delaunay triangulations which originate from the same image element. The Delaunay triangles are removed if the following rule evaluates to be true:

$$\left(\left(C(V_a)=C(V_b)\right)\wedge\left(C(V_b)=C(V_c)\right)\right)\vee\left(\left(C(V_b)=C(V_d)\right)\wedge\left(C(V_d)=C(V_c)\right)\right)$$

where $V_a \neq V_b \neq V_c \neq V_d$ and given a vertex V_x of a Delaunay triangle, $C(V_x)$ is a function that returns the label of the image element upon which the vertex is located.

IV. Segmentation of Characters Connected to Graphics

1. Three Types of Character Connection Problems

In the first author's master thesis work [6], the drawings were drawn according to ANSI drafting standards. All the characters in the drawings were horizontal ones. The connection problem can only occurs on one of four sides of a character. Roughly, there are three types of character connection problems.

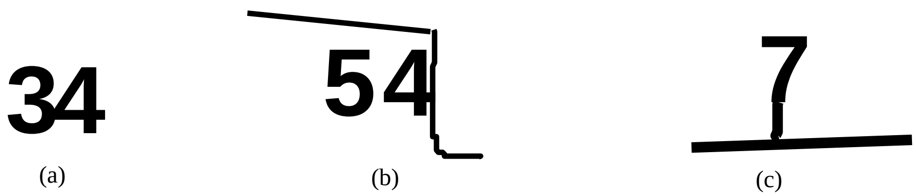


Fig. 4 Typical types of connection problems

I. Character connected to adjacent character, as shown in Fig. 4(a);
II. Among a text string one character connected to graphics, as shown in Fig. 4(b);
III. Single character connected to graphics, as shown in Fig. 4(c).

Among the problem types, a projection analysis method was employed to solve problem type I [6]. A vectorization-based postprocessing method is probably helpful to prob lem type III. This paper is devoted to using area Voronoi tessellation to solve problem type II.

In the following, we first present the current method to segment and group the characters and explain why it fails to detect the characters connected to graphics. Then we give a new grouping method based on area Voronoi tessellation and show how to use it to detect the characters connected to graphics.

2. Current Method to Segment Characters

Size criteria and grouping criteria are used to segment character candidates from graphics in [6]. For each connected component, we construct a bounding box, which tighly encloses the connected component. The size of the most frequently appeared connected components are referred as the average character size. Only the connected components whose bounding boxes' size fit the average character's size are considered as character candidates. For the characters that can be grouped into one text string, the grouping criteria apply: their central points are collinear and the di stance between their bounding boxes fit the desired character spacing, which is determined by the average character size. After the segmentation, a character recognition engine is employed to recognize the characters.

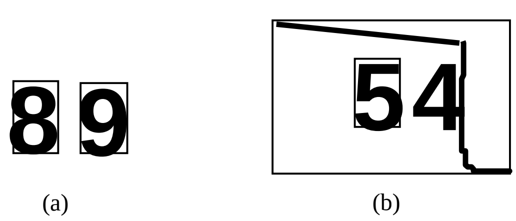
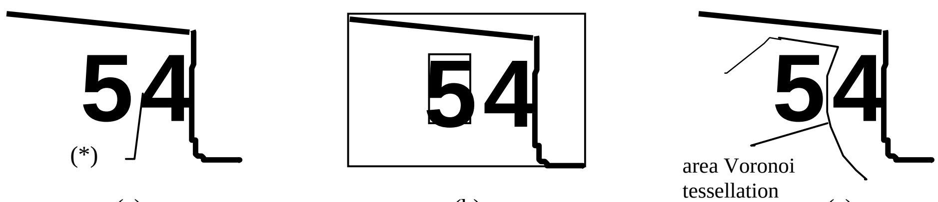


Fig. 5 Examples of using circumscribed rectangle to group text string

For most of the characte rs, their bounding boxes can represent their shapes quite well, as shown in Fig. 5(a). However, for a character connected to graphics, its bounding box is the one that encloses character and the graphics which the character is connected to, as shown in Fig . 5(b), so it can not pass the size criteria and are discarded. Furthermore, if we want to detect them, we have no hint of the existence of such a character.

3. New Method to Detect the Characters Connected to Graphics

To solve problem type II, we should know more about the shapes and relative positions of the connected components. From the definition of area Voronoi tessellation, we can see that the boundary of area Voronoi tessellation of a connected component represents a better shape than its bounding box does. More important, the area Voronoi tessellation can describe the relative distance between adjacent characters more clearly and efficiently. Fig. 6(c) gives an example of area Voronoi tessellation for the Fig. 6(a). It has better shape information than that in Fig. 6(b).



(a) (b) (c)

Fig. 6 Comparison of current method and the new method based on area Voronoi tessellation

Intuitively, in Fig. 6(a), point (*) can give us a hint to identify the connected area of “4” as a character because its distance to the character of “5” fits the standard intra-character spacing. If we use the bounding box to estimate its shape, we cannot know the existence of point (*), as shown in Fig. 6(b). If we use the area Voronoi tessellation, we can make a hypothesis that there is a character near “5” by calculating the distance from “5” to the point (*). According to the grouping criteria, the hypothesis validation for connected characters is processed automatically after the construction of area Voronoi tessellation. Any specific searching process is not necessary.

Given an area Voronoi tessellation on a n engineering drawing, the proposed algorithm consists of two steps.

Step1. Growing up a three-sided box to enclose the connected component in question;

Step2. Locating the cutting position to segment a character connected to graphics and validate if it is a character.

Step1. Growing up a three-sided box to enclose the connected component in question.

The basic approach to locate the potential connected character is similar to the method [4] in some ways. It is to grow up a three -sided box along the free si des, which are not connected to the graphics, of the connected component in question. The open side of the box corresponds to the side in which the character is connected to the graphics. The growing stops whenever a dimension of the box exceeds that of a n average character.

For example, we construct a box from the point on some connected component whose distance to character "5" is close to the average intra -character spacing in a character string. The box encloses the subgraph of the potential characte r as shown in Fig. 7(a). The box is grown up in both x and y directions to cover the connected area. We may have three cases as follows.

- Before the box’s width exceeds the standard character width, the height of the potential charact er is less than the standard character height, as shown in Fig. 7(b). We can assume that the open side is right, then go to step2.1.
- At some position when the box's width is close to the average character width, we get free sides on right and bottom sides but not on top side, Fig. 7(c), (d). Then we can make a hypothesis that the open side is

the top side and go to step2.2. If we cannot get free sides on either bottom or right sides while the box's width exceeds the average character width, the character candidate would be discarded and the procedure ends.

- At some position before the box's width exceeds the standard character width, we get free sides on right and top sides but not on bottom side, Fig. 7(e), (f). Then we can make a hypothesis that the open side is on the bottom side and go to step 2.2. If we cannot get free sides on the right or top sides while the box's width exceeds the average character width, the character candidate would be discarded and the procedure ends.

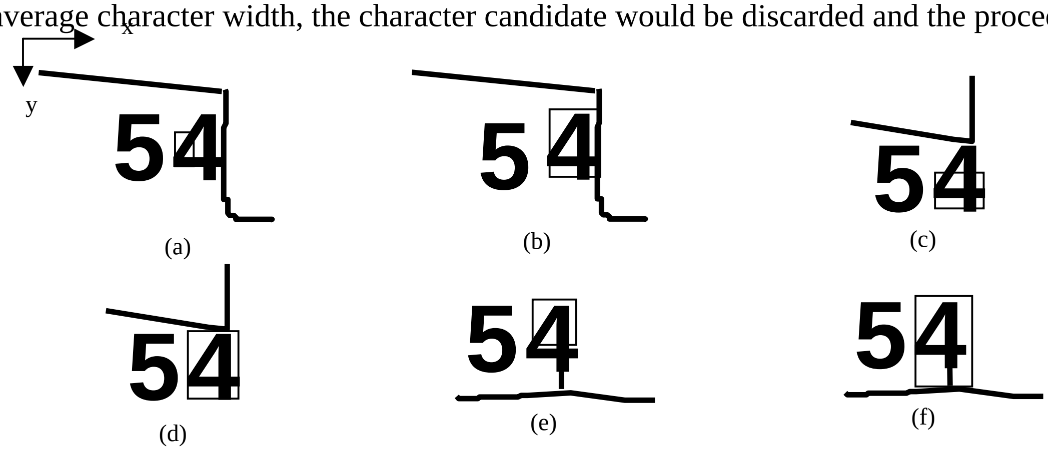


Fig. 7 Illustration of Step 1, in three possible cases

Step 2. Locating the cutting position to segment the character with graphics.

In this step, we test the hypothesis by finding the most possible cutting position with the assistance of a character recognition engine. The basic approach to locate the cutting position is similar to the method in [6], which was used to detach the touching between adjacent characters. Define a projection function of V and its modification function of ϕ. Then we try to detach the connection at the sharp maximum of ϕ and feed the segmented connected component to the character recognition engine. According to different connection scenarios, the projection is constructed in the different directions. We have two different function definitions.

Step 2.1 We define the projection function in y-x direction.

- $V(x)$**:** the function mapping horizontal position to the number of blob pixels in vertical column at that position;
- $\phi(x) = V(x-1) - 2 \times V(x) + V(x+1) \quad (0 < x < Width - 1)$.

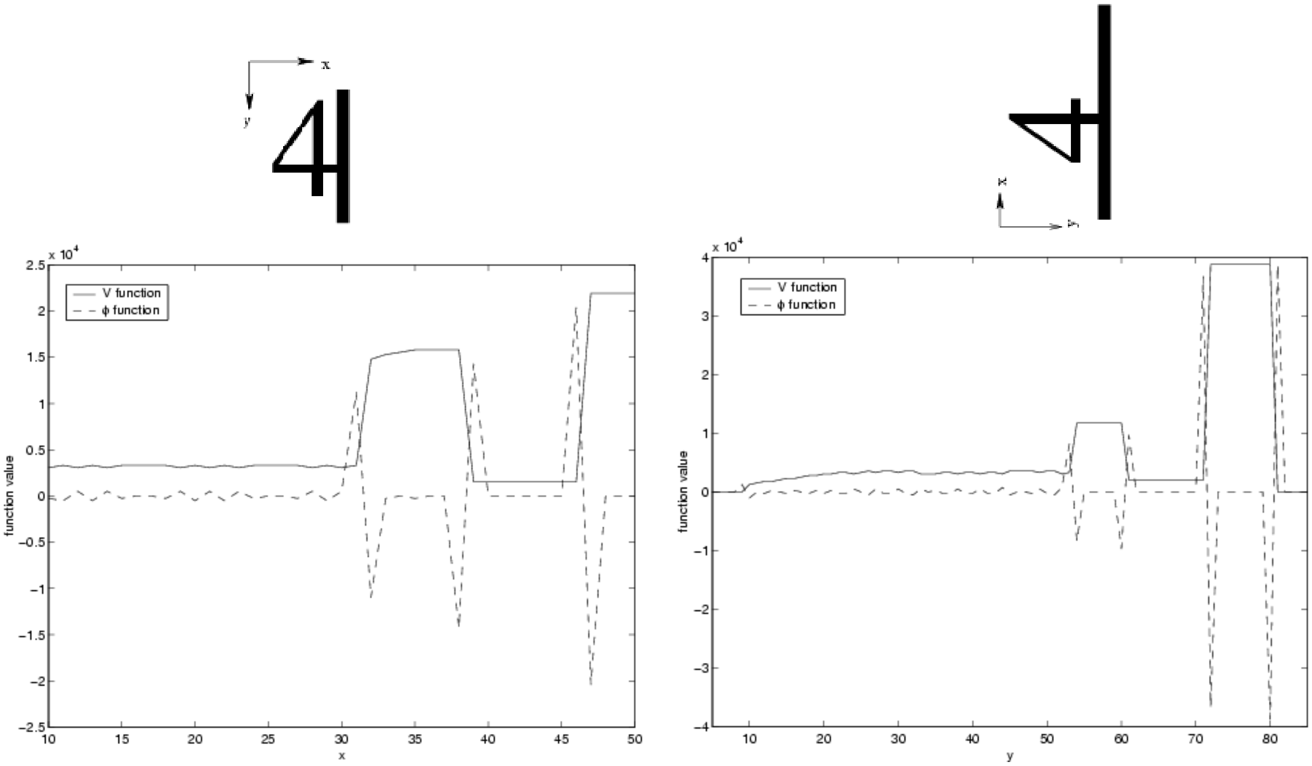
Fig. 8(a) shows one example of the values of function $V(x)$ and $\phi(x)$. We try to detach the character with the graphics in the maximum of $\phi(x)$. Then we use recognition engine to verify whether it is really a touching character. The detection process may test several cutting positions to get the correct character recognition result or discard noncharacter candidate.

Step 2.2 We define the projection function in x-y direction.

- $V(y)$: the function mapping vertical position to the number of blob pixels in horizontal row at that position;
- $\phi(y) = V(y-1) - 2 \times V(y) + V(y+1) \quad (0 < y < Height - 1)$.

Fig. 8(b) shows examples of the function values of $V(y)$ and $\phi(y)$. Note that there are two maximum points in function $V(y)$ in Fig. 8(b). Due to the growing direction, the one closer to the starting point of growing is taken as the cutting position. Other routines are the same with Step 2.1. Just replace x-position with y-position.

The above example algorithm shows how we can find the touching character on the right side or the bottom side of the text string. It can be easily modified to find the touching characters on the left side or top side of the text string. For the touching characters inside a string, similar algorithm can also apply. The new algorithm can completely solve the connection problems type II.



(a) (b)

Fig. 8 (a) Example of functions of V(x) and Φ(x); (b) Example of functions of V(y) and Φ(y).

V. ***Conclusion***

To detect a character connected to graphics is a tough problem. To reach an accuracy rate as high as 98%, special efforts are put by constructing area Voronoi tessellation. Since area Voronoi tessellation represents the shape of connected component better than the bounding box does, it gives us some chance to locate the characters connected to graphics. Projection analysis is used to segment and validate the characters connected to graphics.

From all the above discussion, we demonstrate the correctness and feasibility of the new method to detect the characters connected to graphics. The most time -consuming part is the area Voronoi tessellation construction. Compared with bounding box approach, its extra time complexity is $O(mn \log mn)$, where m is average number of sampled points for a connected area and n is the number of connected areas in the whole drawing. For the detection, the new algorithm also wastes some time on some graphics that are not characters. To gain a 98% or higher accuracy rate, such efforts are worth taking.

Reference

[1]. M. Burge, and G. Monagan. “Extracting Words and Multi -part Symbols in Graphics Rich Document.” In *Technical Report* N^0 *132-95, November 8, 1995*

[2]. A. Nakamura, O. Shiku, M. Auegawa, C. Nakamura, and H. Kuroda. “A method for Recognizing Character Strings from Maps Using Linguistic Knowledge.” In *Proc. Of the Second International Conf. On Document Analysis and Recognition*, pages 561-564, 1993

[3]. R. Casey, D. Ferguson, K. Mohiuddin, and E. Walach. “Intelligent Forms Processing System.” In *Machine Vision and Applicaitons*, Vol. 5, pages 143-155, 1992

[4]. R. Kasturi, S.T. Bow, W. El -masri, J. Shah, J. R. Gattiker and U.B. Mokate. “A System for Interpretation of Line Drawings.” In *IEEE Transactions on Pattern Analysis and Machine Intellignece,* Vol. 12, No. 10, pages 978-992, 1990

[5]. J. Gao, L. Tang, W. Liu and Z. Tang, "Segmentation and Recognition of Dimension Texts in Engineering Drawings", In *IAPR 3rd Int'l Conf. On Document Analysis & Recognition,* Montreal, Canada, pages 528-531, 1995
[6]. Y. Wang, "High-accuracy extraction and recognition of characters in engineering drawings", M.S. thesis, *Tsinghua University,* Beijing, China, Dec. 1996
[7]. A. Okabe, B. Boots, and K. Sugihara. "Nearest neighborhood operation with generalized Voronoi diagrams: a review." In *International Journal of GIS,* Vol. 8, pages 43-71, January-February, 1994
[8]. M. Burge and G. Monagan. "Using the Voronoi tessellation for grouping words and multi-part symbols in document." *Technical Report* N^0 131-95, October 8, 1995
[9]. L. J. Guibas, et al.. "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams." *ACM Trans. Graph.*, Vol. 4, pages 74-123m 1985
[10]. S. Fortune, "A Sweepline algorithm for Voronoi diagrams." *Algorithm*, Vol. 2, pages 153-174