

Constructing Area Voronoi Diagram in Document Images

Yue Lu^{1,2}, Chew Lim Tan²

¹ Department of Computer Science and Technology
East China Normal University, Shanghai 200062, China

² Department of Computer Science, School of Computing
National University of Singapore, Kent Ridge, Singapore 117543
Email: ylu@cs.ecnu.edu.cn, tancl@comp.nus.edu.sg

Abstract

Voronoi diagram of image elements provides an intuitive and appealing definition of proximity, which has been suggested as an effective tool for the description of relations among the neighboring objects in a digital image. In this paper, a fast implementation algorithm is proposed for generating area Voronoi diagram of connected components in document images. A closed convex polygon is utilized to bound each connected component, and the contour is represented using Freeman chain coding, from which we can compute the corresponding Freeman chain coding of its expanded contour directly, without recourse to the operation on pixels. While the contours iteratively expand outwards, the Voronoi diagram is constructed by the intersections of the expanded contours from different connected components. The experimental results show that our proposed approach significantly improves the speed of constructing area Voronoi diagram.

1. Introduction

Voronoi diagram provides a useful tool which is capable of generating a minimal but complete number of neighbors of an element, i.e. only those elements that are closest are obtained, but all are included. The Voronoi diagram of a collection of geometric objects is a partition of space into cells, each of which consists of all the points closer to a particular object than to any others. It divides the continuous space into mutually disjoint subspaces according to the nearest neighbor rule. In the past decades, increasing attentions have been paid to the use of Voronoi diagrams for various applications. For example, there are some reports in the literature about applying Voronoi diagrams to document image analysis in recent years, such as text line orientation detection[1], text region extraction[2], segmenting characters connected to

graphics[3], page segmentation[4], text-line extraction[5], word grouping[6, 7], etc.

In some applications, the connected components in an image are represented by their centroids. An ordinary Voronoi diagram can be generated based on these centroid points using the widely used methods in computational geometry[8]. Voronoi diagrams generated this way are known as *point* Voronoi diagrams. However, such simplification is inappropriate in some cases, because the centroid is a poor representation of shapes for non-round elements. For instance, a document image generally contains various characters of different sizes and different intercharacter gap, the approximation of each element as a single point is too imprecise, and does not adequately represent the spatial structure of the page image. Therefore, the point Voronoi diagram is unsuitable for some applications.

Considering the complex shapes of image elements, the use of so-called *area* Voronoi diagram has been investigated for some applications. However, existing algorithms for generating area Voronoi diagrams are generally time consuming for processing a document image with thousands of connected components. It is therefore imperative to develop a fast implementation algorithm to make it usable for practical applications. For this purpose, a fast algorithm is presented in this paper for the implementation of area Voronoi diagram. The connected components in a document image are bounded using closed convex polygons first. Voronoi diagram is constructed by iteratively expanding outwards the convex polygons, i.e. the intersections of the expanded convex polygons from different connected components produce the Voronoi edges. Our experiments demonstrated the efficiency of the proposed approach.

2. Basic Notions of Voronoi Diagram

For completeness, we first briefly recall some basic definitions of Voronoi diagram. For further details, readers

can refer to [8]. It is noteworthy that we will limit our concerns to only *digitized planar(2-D)* Voronoi diagram in the paper. The most commonly used Voronoi diagram is based on a set of points. Let $P = \{p_1, p_2, \dots, p_n\}$ be a set of points (called *generators* or *sites* hereafter) on a two-dimensional digitized plane \mathcal{Z}^2 , and $d(p, q)$ be the distance between generators p and q . A Voronoi region of a generator p_i consists of all of the points in the plane at least as close to p_i as to any other generators, which is given by

$$V(p_i) = \{p \in \mathcal{Z}^2 | d(p, p_i) \leq d(p, p_j), \forall j \neq i\} \quad (1)$$

Note that, while the Euclidean distance is commonly used in the continuous space, either the Manhattan(city-block) distance or the chess-board distance can be employed in the digitized 2-D space, where the city-block distance is also called the 4-neighbor distance and the chess-board distance is also known as the 8-neighbor distance.

The boundary of its region consists of the equidistant points:

$$\partial V(p_i) = \{p \in \mathcal{Z}^2 | d(p, p_i) = d(p, p_j), \exists j \neq i\} \quad (2)$$

The ordinary point Voronoi diagram generated from a set of generators P is given as a set of Voronoi regions

$$V(P) = \{V(p_1), \dots, V(p_n)\} \quad (3)$$

The Voronoi regions are convex. The boundaries of Voronoi regions are also called Voronoi edges, and the vertices are called Voronoi vertices. The Delaunay triangulation is the geometric dual of the Voronoi diagram. Once the Voronoi diagram for a set of points is constructed it is a simple matter to produce Delaunay triangulation by connecting any two sites whose Voronoi polygons share an edge.

In the above definitions, the generators are isolated points in the plane. In a digital image, in particular a document image containing varying characters of different sizes and different intercharacter gaps, an object or element(connected components) with complex shape and various size generally consists of more than one pixel. The approximation of each element as a single point is too imprecise, and is therefore not adequate to represent the spatial structure and relations among the elements. To use the Voronoi neighborhoods of image elements, it is necessary to generalize the ordinary Voronoi diagram from points to arbitrarily shaped image elements. The ordinary point Voronoi diagram has been generalized in various directions[9], in which the area Voronoi diagram is widely applied in document image analysis.

Given that $E = \{e_1, e_2, \dots, e_n\}$ be a set of distinct elements in the two-dimensional plane \mathcal{Z}^2 , and $d(p, e_i)$ be the distance between a point p and an element e_i defined by

$$d(p, e_i) = \min_{q \in e_i} d(p, q) \quad (4)$$

Then the area Voronoi region $\mathcal{V}(e_i)$ and the area Voronoi diagram $\mathcal{V}(E)$ are defined by

$$\mathcal{V}(e_i) = \{p \in \mathcal{Z}^2 | d(p, e_i) \leq d(p, e_j), \forall j \neq i\} \quad (5)$$

$$\mathcal{V}(E) = \{\mathcal{V}(e_1), \dots, \mathcal{V}(e_n)\} \quad (6)$$

Analogous to the point Voronoi diagram, the boundary of an area Voronoi region $\mathcal{V}(e_i)$ is given by:

$$\partial \mathcal{V}(e_i) = \{p \in \mathcal{Z}^2 | d(p, e_i) = d(p, e_j), \exists j \neq i\} \quad (7)$$

The Voronoi region of a given image element corresponds to a portion of the \mathcal{Z}^2 plane. It consists of the points from which the distance to the corresponding image element is less or equal to the distance to any other image element. In an ordinary Voronoi diagram the boundary shared by two neighboring Voronoi regions is always a segment or a line, while in an area Voronoi diagram it is a curve.

3. Fast Implementation of Area Voronoi Diagrams in Document Images

The area Voronoi diagram is usually constructed from the point Voronoi diagram, by first generating the point Voronoi diagram from the set of points lying on the contours of the elements followed by deleting Voronoi edges from points of the same elements. Generally speaking, this process is time-consuming, especially for an image with a large number of pixels and elements. The processing speed of this algorithm depends on the number of the contours pixels for generating the point Voronoi diagram, and the complexity of the elements for removing the superfluous Voronoi edges and superfluous Voronoi vertices. For convenience, we call it Point Voronoi Based Approach(PVBA).

To accommodate elements in a digital image, an image processing based algorithm for an approximation of a Voronoi diagram has been proposed. Each generator is first replaced with a set of pixels, and then from these pixels the *territories* of the generators are expanded simultaneously at the same speed until the territories collide with one another. This algorithm is straightforward.

Here we pay attention to the latter algorithm, because its results provide the information about relations between elements immediately when the area Voronoi diagram is constructed. In the algorithm described in [8], for each background pixel all of its neighbors are evaluated to decide whether its value should be changed. If all the neighbors are background pixels as well, there is no change. For a worst case, the distance from such a background pixel to its nearest element is D . In the first $(D - 1)$ times of iterative scan, the pixel's value maintains without any change, but the pixel and its neighbors are accessed $9 \times (D - 1)$ times

totally, but a vain attempt. In particular, the processing is quite time-consuming for an image with larger areas of background among the neighboring elements.

In this paper, we modify the algorithm to speed up the procedure of constructing the area Voronoi diagrams in document images. A closed convex polygon is utilized to approximate the contour of a connected component(element) and it is represented using a Freeman chain coding. In each iterative turn, from the Freeman chain coding of a contour we can immediately compute the Freeman chain coding of its expanded contour. The pixels collided by the expanded chain codings of two or more different elements are exactly the Voronoi edge pixels. The following algorithm illustrates the steps of constructing the area Voronoi diagram using our method.

Algorithm-Constructing Area Voronoi Diagram

Step 1: Label connected components to get K connected components. The background pixels are set as the value BACKGROUND_VALUE.

Step 2: Bound each connected component using a closed convex polygon. The value of pixels within the convex polygon is set as the value of its corresponding connected component label $k(k = 1, 2, \dots, K)$.

Step 3: Represent each convex polygon using Freeman chain coding IC_k . The pixels on the contour is set as the value of $k + \mathcal{M}$, where \mathcal{M} is a constant.

Step 4: Initialize the control flag $F_k := true$ of each element $e_k(k = 1, 2, \dots, K)$.

Step 5: If $F_k == false, \forall k \in \{1 \dots K\}$, go to step 6. Otherwise, for each element e_k , if $F_k == true$, then do step 5.1-5.5.

step 5.1: According to IC_k , obtain the Freeman chain coding of its expanded contour EC_k .

step 5.2: For each pixel on the contour IC_k , if its value is not EDGE_VALUE, set its value as k (instead of $k + \mathcal{M}$).

step 5.3: For a pixel on the expanded contour EC_k , if its value is between \mathcal{M} and $K + \mathcal{M}$, it is on the contour of another element. Therefore it is a Voronoi edge pixel, and it is set as EDGE_VALUE. Otherwise, if the value of the corresponding pixel on the expanded contour is BACKGROUND_VALUE, set it as $k + \mathcal{M}$.

step 5.4 If none of the pixels on the expanded contour are background, set $F_k := false$.

step 5.5: $IC_k := EC_k$.

Step 6: Output the resultant image with Voronoi edges.

At the first step, an eight-neighbor connected component analysis algorithm is applied to the input binary image to produce K connected components. The background pixels

are assigned a particular value BACKGROUND_VALUE, say 0 in general. A connected component could be a portion of a character or characters that are touching with each other for a document image. It is noteworthy to mention that if one connected component is encompassed by another one, or if two connected components overlap each other, they are merged to be one connected component. Here, we refer to these connected components as elements, and assume that the elements do not overlap others.

Each of the elements is bounded by a closed convex polygon then. All of the pixels within a bounded convex polygon is set as the value of the corresponding connected component label, i.e. k for the k th element. The contour of convex polygon is represented using a Freeman chain coding, and the pixels on the contour are set as the value of $k + \mathcal{M}$ (note that $\mathcal{M} > K$). Here \mathcal{M} is a constant offset to identify the pixels on the contour from the *interior* pixels.

A processing control flag F_k is initialized as *true* to control the termination condition of iterative operation, i.e. if none of the pixels on an expanded contour of an element are background, the expansion operation of the corresponding element will stop.

In each iterative operation, the contour is expanded outwards with one pixel distance. To effectively and efficiently compute the expanded contours, the Freeman chain coding is applied to represent a contour. In particular, according to the Freeman chain coding of a contour, we can obtain the corresponding Freeman chain coding of its expanded contour, without recourse to the operation on pixels. The details of generating the Freeman chain coding of a contour and computing the Freeman chain coding of the expanded contour are given as follows.

Freeman chain coding is one of the most commonly used methods for efficiently representing a contour. It consists of the coordinate location of a starting point and a sequence of direction codes from one pixel to its neighbor along the contour. Figure 1 gives the eight possible direction codes. In each contour, the most left pixel in the most top run line is considered as the starting point in our approach. Its coordinate location (x_0, y_0) is stored. The sequence of chain codes is obtained by running along the contour in the clockwise direction. As a result, the Freeman chain coding in Figure 2(a) is represented as $\{(4,2) '000000000123443345455570'\}$. It is worth noting that, we duplicate the first chain code to the tail of the coding for convenience in the procedure of computing the chain coding of its expanded contour described below. This does not have any effect on the coding representation.

An important advantage arising from this representation is that, from the Freeman chain coding of a contour we can directly obtain the Freeman chain coding of its expanded contour. The new starting point is immediately located on the upper of the old starting point. For example, the starting point $(4,2)$ in Figure 2(a) is transformed to $(4,1)$ in Figure

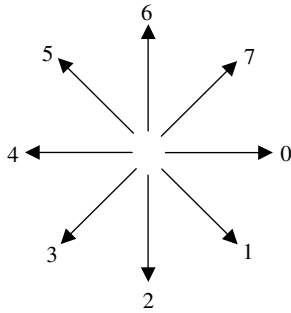
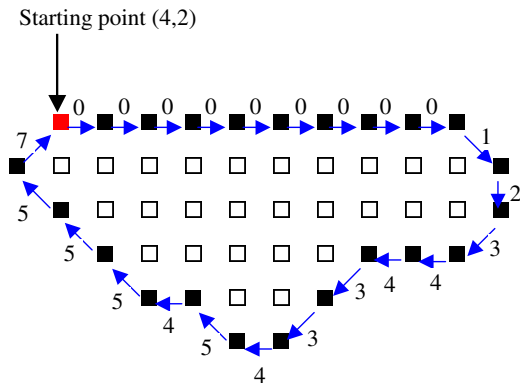
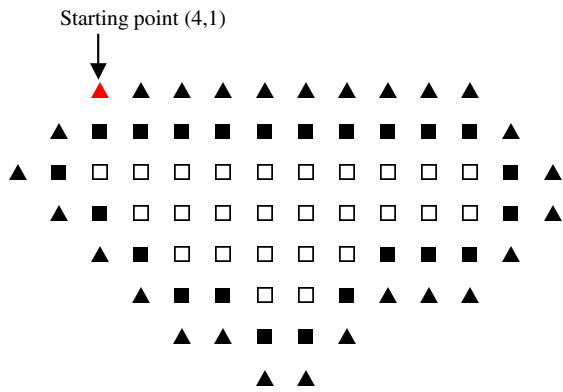


Figure 1. Direction codes



(a)Freeman chain coding



(b)Expanded contour

Figure 2. Freeman chain coding and its expanded contour

2(b). For the transformation of the chain codes, suppose $A = \langle a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots \rangle$ is the Freeman chain coding of a contour, and $B = \langle b_1, b_2, \dots, b_{k-1}, b_k, \dots \rangle$ is the Freeman chain coding of its expanded contour. To produce B for A , a_1 is copied to b_1 first. To consider a_{i+1} ($i \geq$

Table 1. Transform of chain codes

01 \Rightarrow 011	02 \Rightarrow 012	03 \Rightarrow 013
14 \Rightarrow 134	24 \Rightarrow 234	25 \Rightarrow 2355
34 \Rightarrow 334	35 \Rightarrow 3355	36 \Rightarrow 3356
45 \Rightarrow 455	46 \Rightarrow 456	47 \Rightarrow 457
70 \Rightarrow 770	71 \Rightarrow 7711	72 \Rightarrow 7712
60 \Rightarrow 670	61 \Rightarrow 671	10 \Rightarrow 0
07 \Rightarrow 0	43 \Rightarrow 4	54 \Rightarrow 4

1), its preceding code a_i should be taken into account as well. Let's suppose that $A' = \langle a_1, a_2, \dots, a_{i-1}, a_i \rangle$ has been transformed to $B' = \langle b_1, b_2, \dots, b_{k-1}, b_k \rangle$. If $a_i a_{i+1}$ satisfy any one in the leftsides of the arrows in Table 1, b_k is replaced using the rightside of the corresponding arrow. Otherwise, a_{i+1} is copied to b_{k+1} . Based on this processing, the Freeman chain coding of Figure 2(a) will be transformed to $\{(4,1) \langle 000000000112334433454555770 \rangle\}$, which exactly corresponds to the expanded contour in Figure 2(b) represented using the triangles. This process avoids the operation on pixels for generating the expanded contour, which results in a significant improvement of processing speed.

If a pixel on an expanded contour collides with the contour of another element, it is the Voronoi edge pixel. Otherwise, if it is a background pixel, set it as the contour pixel.

4. Experimental Results

To evaluate the validity of our proposed approach, 2158 document images with different sizes, various numbers of pixels and connected components are used to construct their area Voronoi diagram for the purpose of word grouping. The experiments have proven the efficiency of the proposed approach. Figure 3 depicts an example of the Voronoi diagram generated from a document image.

To test the processing speed, an experiment is carried out to compare different approaches to constructing the Voronoi diagrams of the document images. Some examples are listed in Table 2, including point Voronoi based approach(PVBA), Okabe's method and our method. Here the processing time for PVBA consists of sampling points from the contours, generating point Voronoi diagrams and removing the superfluous Voronoi edges and vertices. It can be seen from the comparison that the proposed method is faster than others.

5. Conclusion

The most important and significant contribution of the Voronoi diagram to document image analysis is that it

Table 2. Speed comparison of calculating Voronoi diagram

	image properties			time(in milliseconds)		
	size	# of foreground pixels	# of connected components	PVBA	Okabe's method	our method
image1	884×730	58589	234	315	259	79.5
image2	1280×2056	211836	1798	2632	1115	301.5
image3	1824×3134	328327	1307	2238	2662	413.9
image4	2480×2000	367226	2396	5298	2054	619.5
image5	2592×3300	839130	4064	5417	3162	1134

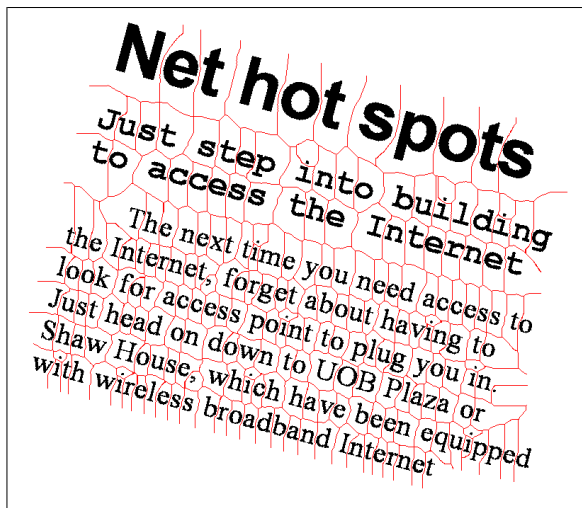


Figure 3. Voronoi diagram of a document image.

introduces neighboring relations into a set of elements (e.g. connected components of characters on a document image). A straightforward but fast and efficient algorithm is implemented in this paper for generating Voronoi diagram of the connected components in document images. A closed convex polygon is utilized to bound the connected component, and the contour is represented using Freeman chain coding, from which we can compute the Freeman chain coding of the expanded contour directly. While the contours expands outwards, the Voronoi diagram is constructed. Experiments show that the speed of our proposed algorithm has been significantly improved, which makes the area Voronoi diagrams be practically implemented in real applications.

Acknowledgements

This work was jointly supported by the National Natural Science Foundation of China (grant No.60475006), and the

Agency for Science, Technology and Research of Singapore (grant No.R252-000-112-202).

References

- [1] D. J. Ittner and H. S. Baird, "Language-free Layout Analysis," *Proceedings of Second International Conference on Document Analysis and Recognition*, Tsukuba, pp.336-340, 1993.
- [2] Y. Xiao and H. Yan, "Text Region Extraction in a Document Image Based on the Delaunay Tessellation," *Pattern Recognition*, Vol. 36 (2003), No. 3, pp. 799-809, 2003.
- [3] Yalin Wang, Ihsin T. Phillips and Robert Haralick, "Using Area Voronoi Tessellation to Segment Characters Connected to Graphics," *Proceedings of Fourth IAPR International Workshop on Graphics Recognition (GREC2001)*, Kingston, Ontario, Canada, September, pp.147-153, 2001.
- [4] K. Kise, A. Sato, and M. Iwata, "Segmentation of Page Images Using the Area Voronoi Diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370-382, June 1998.
- [5] K. Kise, M. Iwata, A. Dengel and K. Matsumoto, "Text-Line Extraction as Selection of Paths in the Neighbor Graph," *Document Analysis Systems*, pp.225-239, 1998.
- [6] M. Burge, G. Monagan, "Using the Voronoi Tessellation for Grouping Words and Multipart Symbols in Documents," *Proceedings of SPIE International Symposium on Optics, Imaging and Instrumentation*, Vol.2573, San Diego, California, pp.116-124, July 1995.
- [7] Y. Lu, Z. Wang, and C. L. Tan, "Word Grouping in Document Images Based on Voronoi Tessellation," *Document Analysis Systems VI, Lecture Notes in Computer Science*, Vol. 3163, pp.147-157, 2004.
- [8] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, *Spatial tessellations: Concepts and applications of Voronoi diagrams(Second Edition)*, Chichester: John Wiley, 2000.
- [9] K. Sugihara, "Approximation of Generalized Voronoi Diagram by Ordinary Voronoi Diagrams," *CVGIP: Graphical Models and Image Processing*, vol.55, no.6, pp.522-531, 1993.