Voronoi Diagram Generation Algorithm based on Delaunay Triangulation

Liping Sun
College of National Territorial Resources and Tourism, Anhui Normal University, Wuhu, China
Email: slp620@163.com

Yonglong Luo
Engineering Technology Research Center of Network and Information Security, Anhui Normal University, Wuhu, China

Yalei Yu
Engineering Technology Research Center of Network and Information Security, Anhui Normal University, Wuhu, China

Xintao Ding
Engineering Technology Research Center of Network and Information Security, Anhui Normal University, Wuhu, China

***Abstract*—Voronoi diagram and its geometric dual, the Delaunay triangulation, both are practical geometric constructions which have been applied extensively in spatial analysis. Considering the low efficiency of the algorithm of indirectly building Voronoi diagram, this paper proposes an improved Voronoi diagram generation algorithm based on Delaunay triangulation of randomly distributed points in the Euclidean plane. In the process of building Delaunay triangulation, correlative edges of points and correlative triangles of edges information is dynamically updated. Theoretical analysis and experimental results show that the proposed algorithm is an efficient method of generating Voronoi diagram.**

***Index Terms*—computational geometry, planar point set, voronoi diagrams, delaunay triangulations**

I. Introduction

The Voronoi diagram is an important branch of computational geometry, which has been proved extremely useful in the algorithmic, geometric and combinatorial problems, such as scientific visualization, isarithmic mapping, spatial process modeling, and so on [1-5]. Consider a set V of n points, the closest-site Voronoi diagram of V subdivides the plane into regions, each region associated with one site $v_i \in V$, and containing all points for which v_i is the closest among all sites of V [6].

In recent years, there are quite a few study of construction algorithm for Voronoi diagram, which can be grouped into two categories: one is direct method, namely directly generating Voronoi diagrams by the set of points, including increment method, divide-and-conquer method, scan-line method [7-9]. Lei et al. [10] presented an improved incremental algorithm for generating Voronoi diagrams. Based on the sweep-line method, a new data structure named right convex hull chain was used to find the location of a new input site in $O(nlogn)$ time. Kalra et al. [11] proposed an incremental algorithm for constructing and reconstructing generalized Voronoi diagrams on grids. Aichholzer et al. [12] gave a simple and practical divide-and-conquer algorithm for generating Voronoi diagrams which was applicable to sites of general shape. Yan et al. [13] designed an efficient algorithm to compute the clipped Voronoi diagrams for a set of sites which were confined to a compact domain. Chen et al. [14] presented a plane sweep algorithm for constructing the Voronoi diagrams using a distance metric, and studied the robustness of the algorithm.

And the other is indirect method, which make use of the relation that Voronoi diagrams is the geometric dual of Delaunay triangulation [15-20]. Mostafavi et al. [21] summarized the related research on Voronoi/Delaunay construction. By insertion and deletion operations in Voronoi diagrams and Delaunay triangulations, the construction process of Voronoi/Delaunay and a variety of applications were described. Sun et al. [22] presented an improved Voronoi generation algorithm based on auto-connected Delaunay triangulation. By the characteristics of ordered target triangles and convex hull, a ray Voronoi diagram was generated by three infinite points. Guibas et al. [23] designed a novel randomized algorithm for the construction of planar Voronoi diagrams and Delaunay triangulations. The efficiency of direct method to build Voronoi diagram is higher than the indirect method, but the data structure of Voronoi diagrams is not easy to describe. Voronoi polygon for each point can be easily obtained by the indirect method, and the data structures of both Voronoi diagrams and

doi:10.4304/jsw.9.3.777-784

Delaunay Triangulation are often required in GIS applications.

Based on the above, a novel Voronoi diagram generation algorithm based on Delaunay triangulation is proposed. Discrete set of points is firstly performed Delaunay triangulation, through point by point insertion method based on convex hull. Then, the information of relevant edges of each point and relevant triangles of each edge is recorded. As a result, the Voronoi diagrams of the planar point set are generated by Delaunay triangulation. The rest of this paper is organized as follows. In Section 2, the related definitions and data structures are presented. The Voronoi diagram generation algorithm based on Delaunay triangulation is fully elaborated in Section 3. Experimental results show the efficiency of the algorithm in Section 4. Finally, conclusions and main findings are given in Section 5.

II. Preliminaries

According to the different construction processes, Delaunay triangulation generation algorithms can be grouped into three categories: triangulation growth, divide-and-conquer algorithm, and point by point insertion method. This paper constructs Delaunay triangulation algorithm which is based on point by point insertion method. Firstly, the set of discrete points of convex hull is establish; Depending on the nature of Delaunay triangulation, triangulation of the convex hull is achieved generate the initial triangulation; Finally, Points inside the convex hull are inserted into initial triangular network to generate the final Delaunay triangulation.

A. The Phases of Algorithm

For a set of distinct point in the Euclidean plane, the Voronoi diagram generation based on Delaunay triangulation algorithm is composed of three main phases.. The first phase constructs the convex hull of the point set. The second phase generates the corresponding Delaunay triangulation. The last phase constructs the Voronoi diagram based on the Delaunay triangulation obtained by the previous phase.

B. Related Definitions and Topological Data Structures

Definition 1 In any triangulation, for point v_0, a triangle edge is called the correlative edge of v_0 iff one of its endpoints is v_0.

Definition 2 In any triangulation, for edge *pq*, a triangle is called the correlative triangle of *pq* iff one of its edge is *pq*.

The data structures implemented in the algorithm as a set of topological relations of points, edges, triangles, and Voronoi diagrams are shown as follows:

```
public struct Vertex          // Mass data points
{
   public double x, y;
    // X & Y coordinates of vertex
   public int isHullVertex;
    // Convex hull vertex tag
   public long[] CorrelativeEdgeID;
    // Correlative edges of vertex
}
public struct Edge            // Edges in triangulation
{
   public long V1Index, V2Index;
    // Edge endpoint's index
   public int isHullEdge;
    // Convex hull edge tag
   public long[2] CorrelativeTriID;
    // Correlative triangles of edge
}
public struct Triangle        // Delaunay Triangles
{
   public long V1Index, V2Index, V3Index;
    // Triangle vertex's index
   public double CircleX, CircleY;
    // X & Y coordinates of circumcircle center
}
public struct Voronoi         // Voronoi diagrams
{
   public Vertex CenterV;
    // Growth center of Voronoi diagrams
   public Vertex[] VArray;
    // Vertex array of Voronoi diagrams
}
```

III. Voronoi Diagram Generation Based on Delaunay Triangulation Algorithm (VDBDT Algorithm)

For the sack of easy presentation of the *VDBDT* algorithm, the relevant symbols are defined as follows: Define arrays *VertexArray*, *EdgeArray*, *TriangleArray*, used to store data points, edges, and triangles, respectively. *HullPoint* denotes the linked list of vertex set of convex hull. For the point set *P*, *Xmin* denotes the set of the points whose *X*-coordinate value is the minimum value, *Xmax* denotes the set of the points whose *X*-coordinate value is the maximum value, *Ymin* denotes the set of the points whose *Y*-coordinate value is the minimum value, *Ymax* denotes the set of the points whose *Y*-coordinate value is the maximum value. Regarding extreme points p_1, p_2, p_3 and p_4, the directed line segment from p_1 to p_2 is denoted $\overrightarrow{p_1p_2}$, the directed line segment from p_2 to p_3 is denoted $\overrightarrow{p_2p_3}$, the directed line segment from p_3 to p_4 is denoted $\overrightarrow{p_3p_4}$, the directed line segment from p_4 to p_1 is denoted $\overrightarrow{p_4p_1}$.

According to above descriptions, the processes of the *VDBDT* algorithm can be presented as follows:

Step 1 : Pretreatment of the planar point set (Figure 1).

Initialize the array *VertexArray* to store the point set *P*. The initial values of *isHullVertex* of all points are 0. Select the point whose *Y*-coordinate value is the minimum value from *Xmin*, denoted p_4. Select the point whose *Y*-coordinate value is the maximum value from *Xmax*, denoted p_2. Select the point whose

X-coordinate value is the maximum value from *Ymin*, denoted p_1. Select the point whose *X*-coordinate value is the minimum value from *Ymax*, denoted p_3. Store the points located on the right side of $\overrightarrow{p_1p_2}$, $\overrightarrow{p_2p_3}$, $\overrightarrow{p_3p_4}$, $\overrightarrow{p_4p_1}$ into array $Array_1$, $Array_2$, $Array_3$, $Array_4$ respectively.

Step 2: Construct the convex hull of the point set *P* based on the improvement of GiftWrapping method (Figure 2). This step can be implemented by the following three operations:

(1) Let *isHullVertex* value of p_1 be 1. Append p_1 to *HullPoint*. Taking p_1 as the starting point, select the point in $Array_1$ which satisfies the condition that the slope of the segment between the point and p_1 is the smallest, denoted c_1. Delete c_1 and the points on the left side of $\overrightarrow{c_1p_2}$ from $Array_1$.

(2) Let *isHullVertex* value of c_1 be 1. Append c_1 to *HullPoint*. Taking c_1 as the starting point, similarly process the other points of $Array_1$, until $Array_1$ be null.

(3) Similarly handle the points of $Array_2$, $Array_3$, $Array_4$. Serially connect the points of *HullPoint* while updating their *RelevantEdgeID*, and append the corresponding edges to *EdgeArray*. The *isHullEdge* value of the edges is assigned the value 1.

Step 3: Generate the Delaunay triangulation based on the convex hull (Figure 3). This step consists of three main operations:

(1) Construct initial triangulation. Each time find two adjacent convex hull edges (sharing a common vertex, denoted v_c) in *EdgeArray*, a third edge, denoted *e*, is then added to connect the free ends of the two convex hull edges, completing a triangle. If the convex hull of any other points is not included in the interior and boundary of the triangle circumcircle, append the triangle to *TriangleArray*, add the third edge *e* to *EdgeArray*, and delete the common vertex v_c from *HullPoint*. Repeat the above operation until *HullPoint* remains three convex hull vertices so far. Connect these three vertices to obtain the last triangle, append the last triangle to *TriangleArray*, and add the new edges to *EdgeArray*.

(2) Insert all non-convex hull vertices of the point set P one by one into the initial triangulation to construct the Delaunay triangulation. Find the triangles in TriangleArray whose circumcircle contains one or more non-convex hull vertices.

(3) Delete the triangles from TriangleArray and the edges shared among triangles from EdgeArray. Connect each non-convex hull vertex to the vertices of the triangle whose circumcircle contains this non-convex hull vertex, append the new triangles to TriangleArray, and add the new edges to EdgeArray.

Step 4: Construct the corresponding Voronoi diagrams. This main process is elaborated as follows (Figure 5):

(1) Choose a point *v* from *VertexArray*. If the *isHullVertex* value of *v* is 1, then go (2). Else go (7).

(2) A correlative edge *ve* is chosen from RelevantEdgeID. If *isHullEdge* value of *ve* is 1, then go (3). Else go (4).

(3) *Q* is the circumcircle center of the triangles whose edges contain *ve*. Draw a vertical ray with *ve*, starting with *Q* in the external direction of the convex hull. Go (5).

(4) Connect the circumcircle centers of the triangles which are on the left and right side of *ve*, a Voronoi edge of *v* can be achieved.

(5) Let *ve* be the next relevant edge of *v*. If *ve* exists, then go (2). Else go (6).

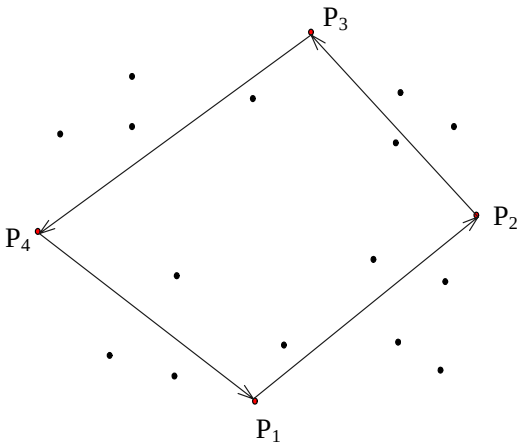
(6) Construct the Voronoi diagram of *v*.

(7) Connect the circumcircle centers of the triangles which are on the left and right side of each relevant edge of *v* to construct the Voronoi diagram of *v*.

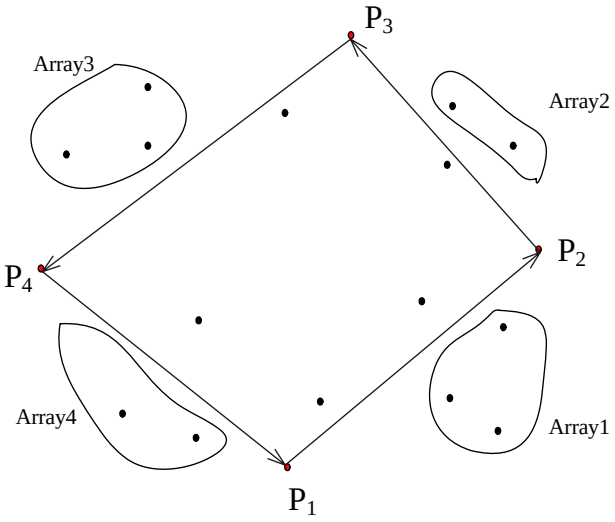
(8) Let *v* be the next point of *VertexArray* to be processed. If *v* exists, then go (1). Else go (9).

(9) End.

(a) Discrete points

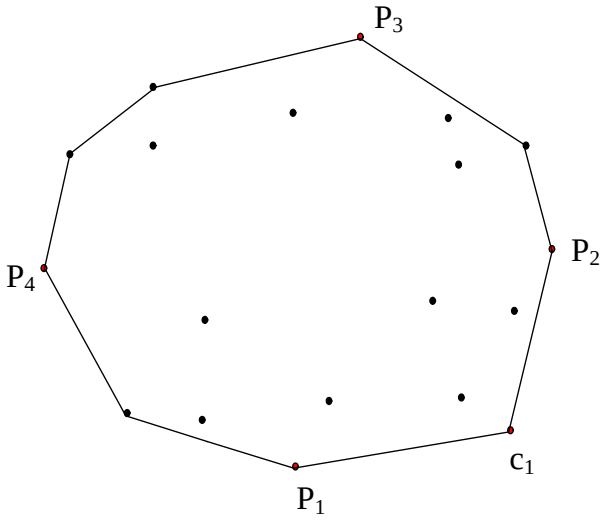


(b) Extreme points p_1, p_2, p_3 and p_4



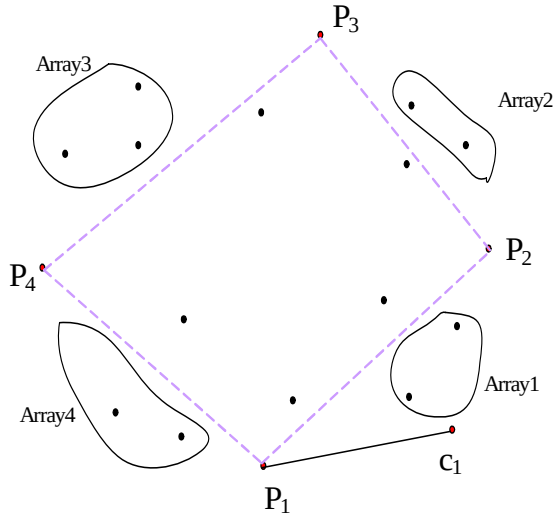
(c) Points classification

Figure 1. Pretreatment of the planar point set.

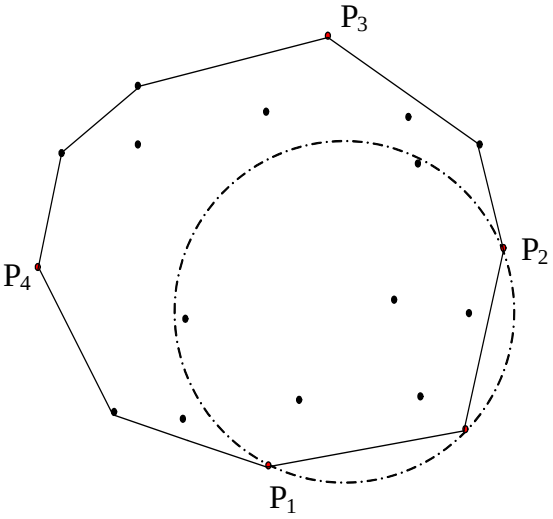


(c) The convex hull

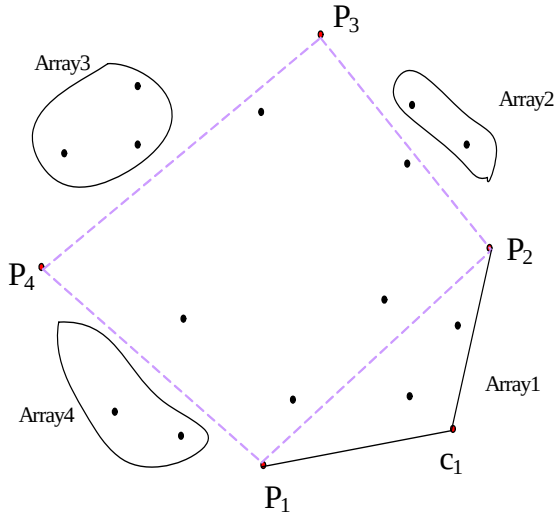
Figure 2. Construct the convex hull of the point set.



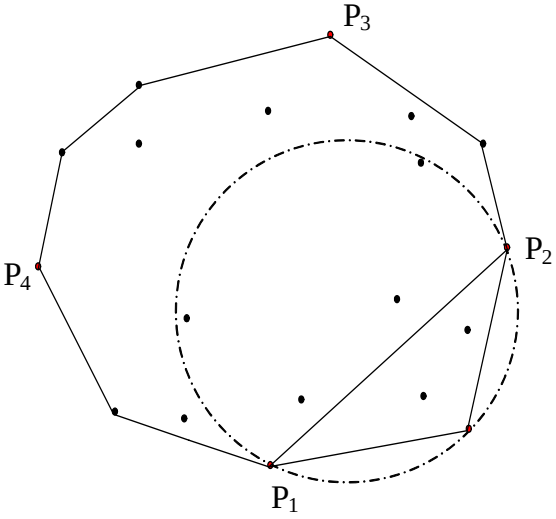
(a) Find c_1



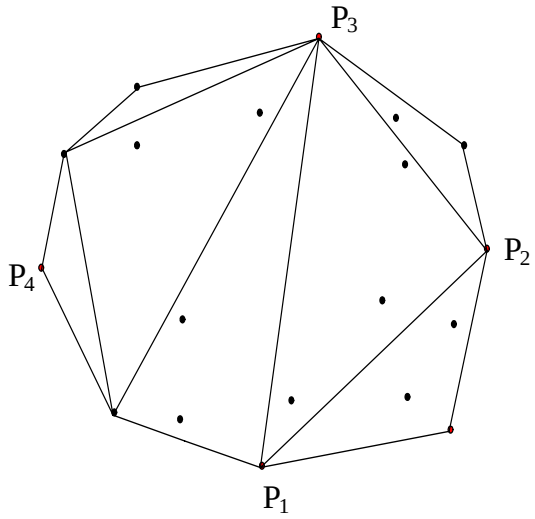
(a) Two adjacent convex hull edges



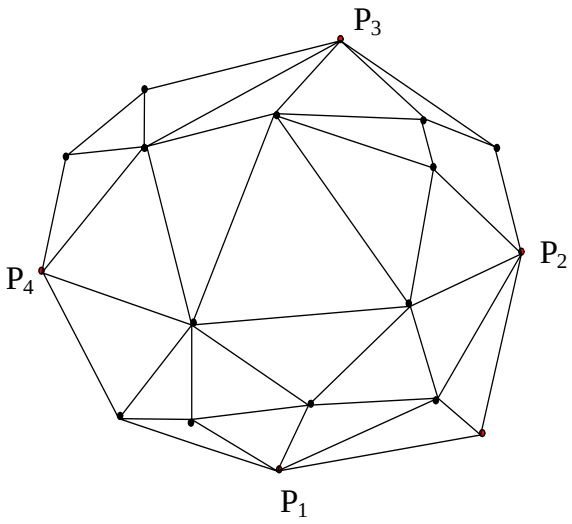
(b) Process the points of $Array_1$



(b) Add an edge to construct a triangle



(c) Construct initial triangulation

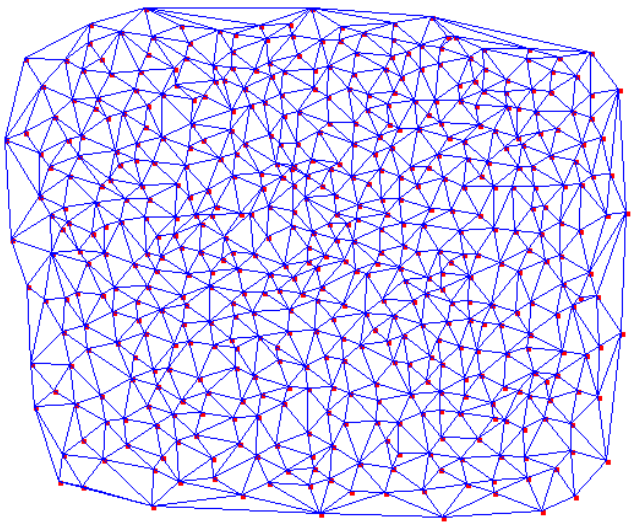


(d) Delaunay triangulation

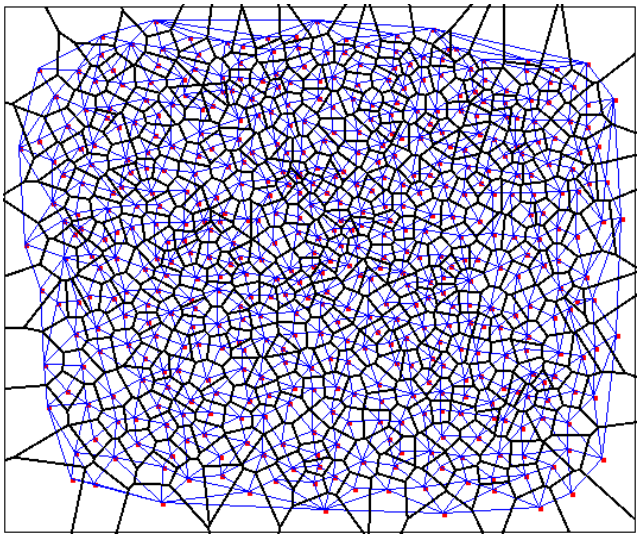
Figure 3. Generate the Delaunay triangulation based on the convex hull.

IV. Experiment Verification

Owing to the fact that Voronoi and Delaunay structures are the duals of each other, the Voronoi diagram generation based on Delaunay triangulation algorithm is presented in the paper. The VDBDT algorithm has been implemented using C# programming language with Microsoft Visual Studio 2010 platform. Randomly generating a set of 500 discrete points, Figure 4(a) depicts the resulting Delaunay triangulation, and Figure 4(b) is the corresponding Voronoi diagrams of the point set.



(a) Delaunay triangulations



(b) Voronoi diagrams

Figure 4. Delaunay triangulations and Voronoi diagrams.

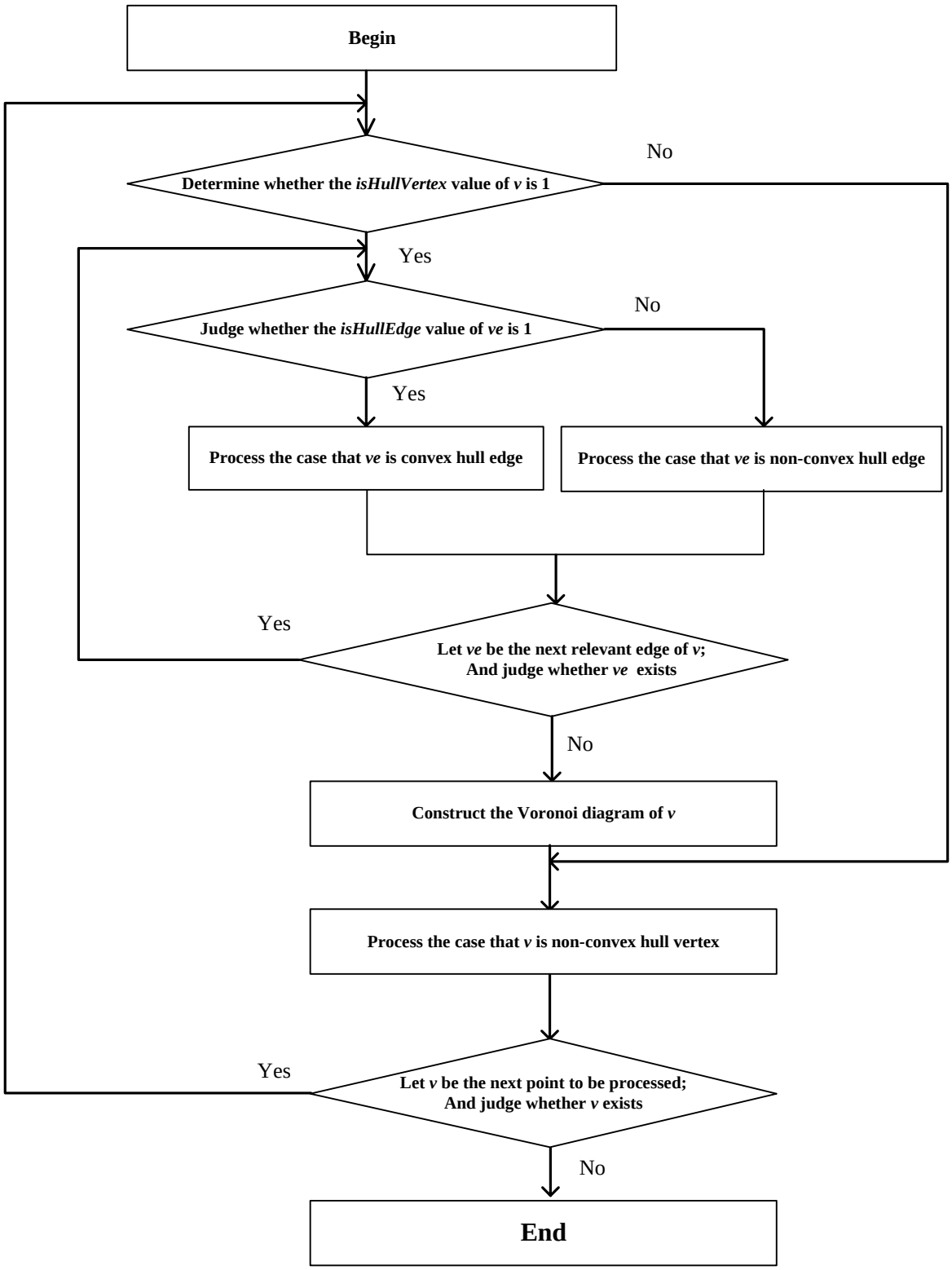


Figure 5. Construct the corresponding Voronoi diagrams based on Delaunay triangulation.

With the purpose of determining the effect produced by the VDBDT algorithm, the relevant experiment has been performed. Various random discrete point set of up to 50,000 are used to construct the corresponding Voronoi diagrams. Figure 6 shows the results of the comparison results between the algorithm of reference [15] and VDBDT algorithm.

The experimental results show that there is no significant difference between the two algorithms when the point number is small. When the point number is large, the computational superiority of the VDBDT algorithm is demonstrated by comparison with reference [15]. The VDBDT algorithm is clear and easy to be implemented. Through this algorithm can get the Voronoi polygon data structure for each destination point, which facilitates its application of GIS spatial analysis.

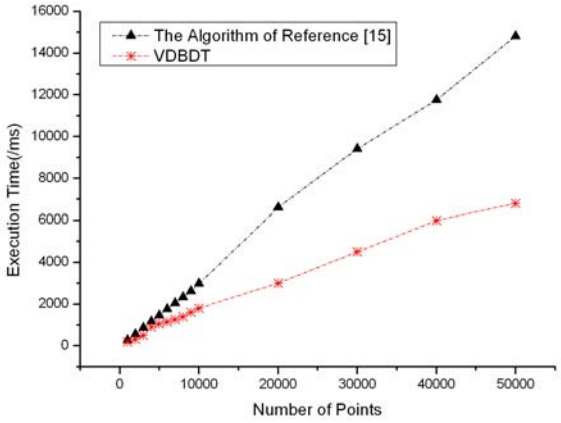


Figure. 6 Comparison results between the algorithm of reference [15] and VDBDT algorithm.

V. CONCLUSION

The VDBDT algorithm divides the set of points into four regions for the generation of the convex hull of the point set. Non-convex hull vertices are gradually excluded in the process of searching convex hull vertices, as a result that number of comparisons of slope is reduced. In the process of constructing a Delaunay triangulation, correlative edges of points and correlative triangles of edges information is dynamically updated. First get all correlative edges of each point, and then find the correlative triangles of the correlative edges. Voronoi polygon of each discrete point can be achieved by circumcircle center of the correlative triangles. Experimental results show the efficiency of the algorithm.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (No. 61370050) and Natural Science Foundation of Anhui Province (No. 1308085QF118).

REFERENCES

[1] Hongzhi Liu, Zhonghai Wu, D. Frank Hsu, Bradley S. Peterson, Dongrong Xu, "On the generation and pruning of skeletons using generalized Voronoi diagrams," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2113-2119, 2012

[2] Darius Geiß, Rolf Klein, Rainer Penninger, Günter Rote, "Optimally solving a transportation problem using Voronoi diagrams," *Computational Geometry*, vol. 46, no. 8, pp. 1009-1016, 2013

[3] Ioannis Z. Emiris, Angelos Mantzaflaris, Bernard Mourrain, "Voronoi diagrams of algebraic distance fields," *Computer-Aided Design*, vol. 45, no. 2, pp. 511-516, 2013

[4] Tamal K. Dey, Lei Wang, "Voronoi-based feature curves extraction for sampled singular surfaces," *Computers & Graphics*, vol. 37, no. 6, pp. 659-668, 2013

[5] Arman Didandeh, Bahram Sadeghi Bigham, Mehdi Khosravian, Farshad Bakhshandegan Moghaddam, "Using Voronoi diagrams to solve a hybrid facility location problem with attentive facilities," *Information Sciences*, vol. 234, pp. 203-216, 2013

[6] Aurenhammer F, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345-405, 1991

[7] Nancy M. Amato and Edgar A. Ramos, On computing Voronoi diagrams by divide-prune-and-conquer. In Symposium on Computational Geometry, 1996, pp. 166-175

[8] M Evazi, H Mahani, "Generation of Voronoi grid based on vorticity for coarse-scale modeling of flow in heterogeneous formations," *Transport in Porous Media*, vol. 83, no. 3, pp. 541-572, 2010

[9] Wenbin Zhao, Zhengxu Zhao, "Research on Engineering Software Data Formats Conversion Network," *Journal of Software*, vol. 7, no. 11, pp. 2606-2613, 2012

[10] MENG Lei, ZHANG Junwei, WANG Xiaoting, YANG Chenglei, "An improved incremental algorithm for Voronoi diagram," *Journal of Image and Graphics*, vol. 15, no. 6, pp. 978-984, 2010

[11] Nidhi Kalra, Dave Ferguson, "Anthony Stentz. Incremental reconstruction of generalized Voronoi diagrams on grids," *Robotics and Autonomous Systems*, vol. 57, no. 2, pp. 123-128, 2009

[12] Oswin Aichholzer, Wolfgang Aigner, Franz Aurenhammer, Thomas Hackl, Bert Jüttler, Elisabeth Pilgerstorfer, Margot Rabl, "Divide-and-conquer for Voronoi diagrams revisited," *Computational Geometry*, vol. 43, no. 8, pp. 688-699, 2010

[13] Dong-Ming Yan, Wenping Wang, Bruno Lévy, Yang Liu, "Efficient computation of clipped Voronoi diagram for mesh generation," *Computer-Aided Design*, vol. 45, no. 4, pp. 843-852, 2013

[14] Zhenming Chen, Evanthia Papadopoulou, Jinhui Xu, "Robustness of *k*-gon Voronoi diagram construction," *Information Processing Letters*, vol. 97, no. 4, pp. 138-145, 2006

[15] Victor J D Tsai, "Fast topological construction of Delaunay triangulations and Voronoi diagrams," *Computers & Geosciences*, vol.19, no. 10, pp. 1463-1474, 1993

[16] Christian Sohler, "Fast reconstruction of Delaunay triangulations," *Computational Geometry*, vol. 31, no. 3, pp. 166-178, 2005

[17] S Zhu, K Lin, Z Zeng, L Liu, W Hong. A Sampling Method Based on Gauss Kernel Learning and the Expanding Research," *Journal of Computers*, vol. 7, no. 2, pp. 547-554, 2012

[18] Jonathan Quinn, Feng Sun, Frank C. Langbein, Yu-Kun Lai, Wenping Wang, R. Martin, "Improved initialisation for centroidal Voronoi tessellation and optimal Delaunay triangulation,"*Computer-Aided Design*, vol. 44, no. 11, pp. 1062-1071, 2012
[19] Y Zhao, S Liu, Y Zhang, "Spatial Density Voronoi Diagram and Construction," *Journal of Computers*, vol. 7, no. 8, pp. 2007-2014, 2012
[20] Y Yao, S Ning, M Tian, W Yang, "Privacy-preserving Judgment of the Intersection for Convex Polygons," *Journal of Computers*, vol. 7, no. 9, pp. 2224-2231, 2012
[21] Mir Abolfazl Mostafavi, Christopher Gold, Maciej Dakowicz, "Delete and insert operations in Voronoi/Delaunay methods and applications," *Computers & Geosciences*, vol.29, no. 4, pp. 523-530, 2003
[22] Jizhong Sun, Yan Hu, Yong-qiang Ma, "Voronoi diagram generation algorithm based on Delaunay triangulation," *Journal of Computer Applications*, vol. 30, no. 1, pp. 75-79, 2010
[23] Leonidas J Guibas, Donald E Knuth, Micha1 Sharir, "Randomized incremental construction of Delaunay and Voronoi diagrams," *Algorithmica*, vol. 7, no. 4, pp. 381-413, 1992

Liping Sun She is currently an associate professor of the Department of Computer Science and Technology at Anhui Normal University. Her research interests include computational geometry, computer modeling and spatial data mining.

Yonglong Luo He is currently a professor of the Department of Computer Science and Technology at Anhui Normal University. His research interests include information security, applied mathematics and computational geometry.

Yalei Yu He is a master, whose main research interests include information security and applied mathematics.

Xintao Ding He is currently a PhD candidate, whose main research interests include information security and applied mathematics.