Proved Quick Convex Hull Algorithm for Scattered Points

Linna Huang
Department of Computer Engineering
Cangzhou Normal College
Cangzhou, China
Hln0322@163.com

Guangzhong Liu
Department of Computer Engineering
Hebei Engineering and Technical College
Cangzhou, China
czlgzh@heinfo.net

***Abstract*—On the basis of the concept of right-shell-tree and left-shell-tree, a fast convex hull algorithm for scattered points based on a binary tree is put forward. It can dynamically reduce the scale of scattered point rapidly by removing a number of non-convex hull vertexes while finding each convex hull vertex. Normally, it can achieve linear time complexity. The algorithm obviates the judging process of the convex hull vertices' connected relation. It is applicable to any complex scattered points, and simple and easy to achieve.**

Keywords- computer application; convex hull; binary tree; algorithm

I. INTRODUCTION

In the application of GIS (Geographic Information System), the original data is frequently a number of discrete data, such as the data of rainfall distribution, the groundwater depth, the water level measurement point of flood. In considering different order of data collection, transmission, and input, the data record is scattered and disordered in general, called set of scattered points.

The convex hull of quickly building set of scattered points data is the primary problem that needed to be solved through dynamic calculation of area, regional crop, the subdivision of Triangulated Irregular Network (TIN), computer-aided design and some other problems [1]. Therefore, it leaded to extensive researches by domestic and foreign scholars, and it proposed a variety of convex hull algorithm. Among them, what is more classic are Graham scanning method, Jarvis stepping method, wrapping method, dividing and conquering method, and son on [2], some other algorithms [3] [4] are improved classic algorithms. It effectively increased the efficiency of convex hull generation. However, during practical application, it still cannot meet the requirement of GIS dynamic simulations in real-time, and consequently faster convex hull algorithms are necessary.

In general, two issues need to be addressed during the construction of convex hull: firstly, it needs to find out vertices of convex hull from a large number of discrete points; secondly, the connections between these vertices need to be determined [5]. Yao [6] has proved that the level of time complexity required in solving these two problems is O (nlogn) at least (among them, n is the number of discrete points), the level of time complexity using Graham algorithm as representative based on ordered convex hull algorithm can reach the lower limit. In fact, under normal circumstances, convex hull vertices are only a small part of the set of points. Most of the points are within the convex hull. Apparently, through removing points outside the convex hull as much as possible, it could reduce the size of point set in order to achieve the purpose of improving the efficiency of the algorithm. This is so-called quick convex hull technology [4]. Therefore, Floyd quadrilateral method and octagonal method as reference [4] came out. Their common feature is that it could increase the pretreatment of point set and consequently eliminate a part of internal points within convex hull in advance. Although this approach still failed to reduce the time complexity of algorithm, it in did improve the efficiency in many cases.

However, no matter the quadrilateral method or octagonal method there are shortcomings in three aspects at least: 1) Removing points inside the convex hull can only be used for once in initial. It cannot remove non-convex hull vertices dynamically. Actually, that it cannot achieve "as much as possible". 2) When xmax and ymax is the same point, at the same time, xmin and ymin is also the same one point, it cannot constitute the initial quadrilateral or octagonal. 3) When the set of points unevenly distributed and most of the points located in the edge of the case, the efficiency of the algorithm is significantly reduced.

According to the concept of quick convex hull, this paper presents an algorithm based on binary tree that builds convex hull with scattered point. It inherits fast and convenient these features of binary tree. It not only applies to a variety of point set distributions, but also removes non-convex hull vertices repeatedly and dynamically. So that, the scale of candidate point sets is narrowed quickly, meanwhile it also saves the meaningless operation in ordering.

II. THE ESTABLISHMENT OF BINARY TREE

A. Basic concepts

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

Definition 1: A binary tree is a finite set with n ($n \geq 0$) nodes, where there is one and only one node point without parents, called as "root" node. Each node has two "children" in maximum, the node on the left side called "left child", the

978-1-4673-1411-4/12/$31.00 ©2012 IEEE

node on the right side called "right child". The node without child is called "leaf" node. A node with only one child is called "single branch" node. A node with only a "left child" is named as "left single branch" node, while it with only a "right child" is named "right single branch" node. Nodes with two children are called intermediate nodes.

Definition 2: Within point set P, line Lab determined by any two points a,b divided P into two independent sub-sets, shown in Figure 1. A binary tree that all generated by the sub-set on the right side of directed edge $\overline{ab}$ is called the right shell tree; while, the binary tree all generated by the sub-set on the left side of directed edge $\overline{ab}$ is called the left shell tree.

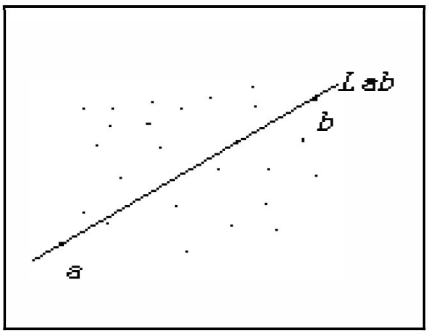

Figure 1. Two independent sub-sets of P

Theorem 1: Right shell trees with counterclockwise directed edges are equivalent to left shell trees with clockwise directed edges.

Proved As shown in Figure 2, if directed edges are counter-clockwise traversed, sub-scattered point sets on the right side of directed edges $\overline{ac}$, $\overline{cb}$, $\overline{ba}$ they are all outside the initial convex hull. While, if directed edges are clockwise traversed, sub-scattered point sets on the left side of directed edges $\overline{ab}$, $\overline{bc}$, $\overline{ca}$ they are all outside the initial convex hull. Sub-scattered point sets on the left side of a certain directed edges (such as $\overline{ab}$) is same to sub-scattered point sets on the right side of $\overline{ba}$. Therefore, the theorem is proved.

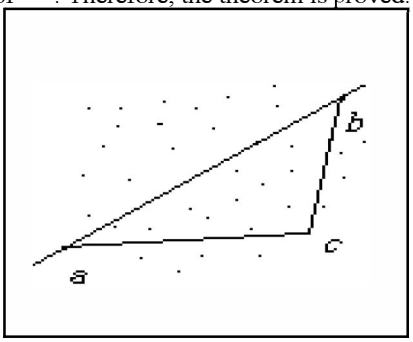

Figure 2. The farthest point to $\overline{ab}$ and interior points of convex hull

Theorem 2: If the point a, b are vertexes of convex hull, and there is one point c in a sub-scattered point set, moreover the distance from c to the line Lab determined by a,b is the farthest, c must be a convex hull vertex.

Theorem 3: Points inside the range determined by $\triangle$abc must be interior points of the convex hull.

According to Theorem 3, it could gradually exclude some candidate vertexes of the convex in order to reduce the scale of scattered point set quickly.

B. Node structure

Each node of binary tree is composed by three vertex data fields and two pointer fields, as shown in Figure 3. Defined as follows:

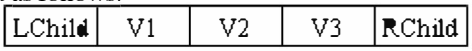

Figure 3. Node data structure

```
Struct TreeNode
{
POINT V1, V2, V3; // Convex hull vertex
TreeNode *LChild, // Left pointer
TreeNode *RChild; // Right pointer
}
```

Among them, pointer LChild points to the node composed by v1, vd the furthest point away from $\overline{v_1v_2}$, and v2. If vd does not exist, it points to NULL. RChild points to the node composed by v2, vd the furthest point away from $\overline{v_2v_3}$, and v3. If vd does not exist, it points to NULL.

C. Binary tree structure

According to Theorem 1, only taken right shell tree as an example to describe the construction process of binary tree.

To assume that there are scattered point set P = {pi}, i = 1,2, ..., n (n is the number of scattered points).

1) Guidelines of root nodes.

Selection of root node is the key influencing efficiency of algorithm. Rule one: the point that generates root node must be on convex hull, ie convex hull vertices. Rule two: the resulting binary tree must always be a balanced binary tree. In order to meet these two rules above, it has to select extremum distributed scattered point set as equal as possible as the root node (Figure 1).

2) Generation of root node.

It needs to find out the minimum point of y coordinate in P. If there are multiple points, the minimum point of x coordinate needs to be chosen, marked as "1". At the same time, it needs to find out the maximum point of y coordinate in P. If there are multiple points, the maximum point of x coordinate needs to be chosen, marked as "2". Obviously, points "1" and "2" are convex hull vertexes, generating root node "121". Then, link all the scattered points on the right side of "12" to the left pointer of node "121" and link all the scattered points on the right side of "12" to the right pointer of node "121".

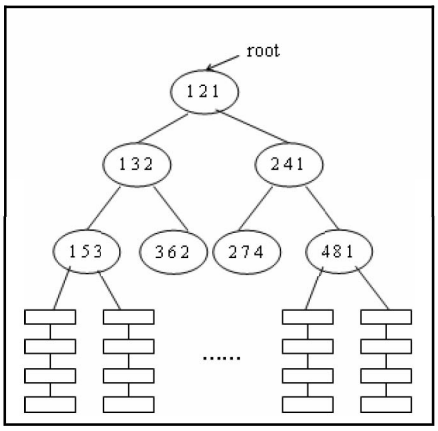

Figure 4. Generation of binary tree

3) Generation of the other nodes.

Find out the point has the furthest distance from the edge "1 2" in the table of scattered point chain pointed out by LChild, marked as "3". Generate a new node "132", and link it to the left pointer of node "121" (Figure 4). Meanwhile, link the scattered point on the right side of "1 3" to the left pointer of node "132". Link the scattered point on the right side of "3 2" to the left pointer of node "132". Take the same processing to node "132" until the farthest point does not exist in the table of chain.

Using the same rules to process the root node RChild.

D. Method to calculate the distance from point to edge

As we all know, the cost to calculate distance from point to edge is relatively large. However, the purpose of calculating distance from points to edge is to compare which point is farther in distance from the given edge in chain table. Actually, there is no need to calculate this distance. Consequently, this paper using a "pseudo-distance" technique succeeded in reducing the calculating cost. Taking Figure 2 as an example to explain the calculating method of pseudo-distance.

$$PD = \frac{CP}{SD} \qquad (1)$$

Where: PD is pseudo-distance; CP is the absolute value of cross product of (b-a) and (c-a); SD is sum of squares of a, b coordinate difference.

It can be seen from equation (1), it at least avoided the most costly square root calculation.

III. The Searching of Convex Hull

The edge of convex hull only stored in leaf node and single branch node. Using the following algorithm to find out convex hull.

1) If it is a leaf node, $\overline{v_1v_2}$ and $\overline{v_2v_3}$ are two edges of the convex hull.

2) If it is a left single branch node, $\overline{v_2v_3}$ is an edge of the convex hull.

3) If it is a right single branch node, $\overline{v_1v_2}$ is an edge of the convex hull.

IV. Algorithm Analysis

A. Time efficiency analysis

1) The worst case of time complexity

The worst case of this algorithm exists when the number of convex hull vertices is equal to the number of scattered points. Then division of each level will not actually reduce the size of point set. Setting the number of scattered points is n, to find out the initial two extremum to compare for n-1 times in maximum. The division in first time needs n-2 times. If these n-2 scattered points they all belong to a sub-tree, finding the farthest point only needs a maximum of n-3 times. Still assuming that these n-3 scatter belong to a sub-tree, finding the farthest point needs to take n-4 times. Following this, until the farthest point is 0. Then, it generates a single branch binary tree. Its time complexity is:

$$(n-1)+(n-2)+\ldots+1 \le n2$$

$$T1 = O(n2) \qquad (2)$$

Since the binary tree generated is actually a linear form, the time complexity of convex hull finding is:

$$T2=O(n) \qquad (3)$$

So, the total time complexity in the worst case as follows:

$$T=T1+T2=O(n2) \qquad (4)$$

2) The time complexity under common situation

In fact, the worst case above does not exist. According to the constructing method in 2.3, it could generate a generally balanced binary tree. Further, each level of division could always exclude part of the points of non-convex hull vertices. To avoid losing generality, assume that the final convex hull has m vertices. For each time of division, at least one convex hull vertex could be found.

So, times of searching would not exceed m * n, so the time complexity is:

$$T1=O(m*n) \qquad (6)$$

The time complexity of searching convex hull in a binary tree with k nodes:

$$T2=O(k*\log k) \qquad (7)$$

Total time complexity is:

$$T=T1+T2= O(m*n)+O(k\log k) \qquad (8)$$

Generally, m, k are much smaller than n. Formula (8) could achieve O(n) approximately, which can be verified from Table 2.

B. Space efficiency analysis

In order to facilitate the storage of connection between convex hull vertexes, it uses redundant data structure. The root node has one redundant data, while the other remaining nodes have two redundant data. In binary tree with k nodes, it needs redundant storage space as follows:

$$S = \sum_{i=2}^{k} 2^{i+1} + 1 \qquad (5)$$

It can be seen from formula (5), the auxiliary space needed is comparatively large. In fact, with the rapid development of storage technology, the storage space is no

longer an issues of concern. Instead, the speed have become increasingly demanding. This algorithm trades space for time.

C. *Test of numeral value*

The algorithm of this paper does tests by using random coordinate points and programming with VC++ 6.0,which leads to the results of Table 1 and Table 2.The sort method of Graham is a quick sort while Jarvis algorithm is done with Robert C.Martin's（Uncle Bob）false-angle incremental approach.

TABLE I. STATISTICS OF POINT-GROUP REDUCTION AFTER DIVISION ON EACH LAYER

	point-group	tree height	node number
original data	10000	0	1
1st layer division	3040	1	3
2nd layer division	73	2	7
……	…	…	…
results	0	5	14

TABLE II. PERFORMANCE COMPARISON WITH OTHER ALGORITHM (MS)

data amount	convex hull sides	method of this paper	Graham method	Jarvis method	Gift wrapping method
1000	14	8.6	22.5	10.9	12.6
5000	23	43.0	200.2	78.1	89.5
10000	27	81.9	611.1	177.8	189.6
50000	21	374.8	7759.8	921.4	1043.1
100000	29	966.1	11651.0	2514.3	2140.4

From Table 1, under the condition of data amount of 10000, the only tree height 5 can complete the structure of the whole convex hull. After each layer's division, points-group reduces quickly with great speed of convergence.

Table 2 shows the time needed by the algorithm from this paper vary linearly. With the increase of n--scatterplot quantity, whose algorithm speed is superior to others greatly.

V. CONCLUSION

Quick convex hull algorithm is put forward on the base of binary tree with the foundation of the right-shell-tree (left-shell-tree). This algorithm has the following features:

(1) Construct the convex hull directly by using scattered data without sorting.

(2) During constructing the convex hull, eliminate continuously the vertexes of non- convex hull to reduce the point-group quickly, which can lead to usual linear time complexity.

(3) This algorithm is applied for various point-set distribution, avoiding effectively algorithm failure.

(4) Make good use of binary tree to avoid movement of large data.

(5) Because of the stored directed edge, the complete convex hull can be pictured without judging the connection between vertexes.

REFERENCES

[1] Li Junhui and Li Ziyang, "A Quick Algorithm and Programming to Determine Convex Hull for Planar Scattered Point Set in GIS," Journal of Beijing Union University(Natural Sciences), vol. 23, pp. 32–34, March 2009.

[2] Zhou Peide, Design and Analysis of Geometric Algorithm, Second Edition, Beijing: Tsinghua University Press, 2005.

[3] Wu Wenzhou, Li Lifan and Wang Jiechen, "An Improved Graham Algorithm for Determining the Convex Hull of Planar Points Set," Science of Surveying and apping, vol. 35, pp. 123–125, June 2010.

[4] Yu Xiangyu, Sun Hong and Yu Zhixiong, "Improved Rapid Calculating Method of Two-dimension Point-group Convex Hull," Journal of Wuhan University of Technology, vol. 27, pp. 81–83, October 2005.

[5] Zhang Xiangjing, "Research of Constructing Convex Hull Algorithm Based on Scattered Points," Journal of Geomatics, pp. 9–12, April 2000.

[6] Yao A C C, "A Lower Bound to Finding Convex Hulls," Journal of the ACM , vol 28, pp. 780–787, 1981.