An Efficient Algorithm of Convex Hull for Very Large Planar Point Set

Guangquan Fan
School of Management Science and Engineering
Hebei University of Economics and Business
Shijiazhuang, China
fanguangquan@126.com

Liping Ma
Computer Center
Hebei University of Economics and Business
Shijiazhuang, China
mlpbox@126.com

Bingru Yang
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China
bryang_kd@yahoo.com.cn

***Abstract*—In the paper, we present and prove Castle Theorem of Convex Hull, design and realize the Fast Rampart Searching Algorithm. The algorithm can be treated as the preprocess of Convex Hull calculation of very large planar point set. When calculating the Convex Hull of a very large planar point set, we can use the Fast Rampart Searching Algorithm to get a very small part points as candidate point set, and then we can get the Convex Hull of the whole planar point set from the candidate point set through other algorithms.**

Keywords-fast rampart searching algorithm; castle theorem; convex hull; computational geometry

I. INTRODUCTION

Convex Hull is the smallest convex set that contains all the points of the set S. Hereinto, the convex hull of a planar point set is the most important and foundational problem. The convex hull problem of a set of planar line segments or planar polygons can be transformed into the convex hull problem of a planar point set. The convex hull of planar point set S is the smallest convex polygon that contains all the points of S, and its vertexes belong to S.

The convex hull is one of the most prevalent and foundational structure, and takes an important role in Computational Geometry[1]. It not only has many specialties, but also is an efficient tool in constructing other Geometric figures. Many practical application problems can come down to Convex Hull problem, it has comprehensive application in Image Processing, Pattern Recognition and Computer Graphics etc [2,3,4].

At present, the main algorithms for the convex hull of a planar point set are: Gift Wrapping, Graham scan algorithm, Divide and Conquer algorithm, incremental algorithm, real-time algorithm, Quick Hull algorithm and Z3-8 algorithm (designed by Peide Zhou)[5] etc. Because of the large amount of spatial queries, especially when the number of points is very large in the point set, the efficiency of these algorithms becomes so poor that they can not satisfy the demands of the practical applications.

This paper presents and proves the Castle Theorem of the convex hull, designs and realizes the Fast Rampart Searching Algorithm (FRSA). The algorithm can be treated as a data preprocess of convex hull calculation of very large planar point set. When calculating convex hull of very large planar point set, we can first use this algorithm to get a small number of points as a candidate point set. Then, we can get the convex hull of whole point set quickly from the candidate point set through other convex hull algorithms.

II. CASTLE THEOREM

A. Basic thought

According to the definition of convex hull, the calculation of convex hull (if there is no special explanation, the convex hull indicates the convex hull of a planar point set in this paper) involves only a small part of the points which are located at the exterior of the point set, and others are irrelevant. If we can get those exterior points which influence the calculation of the convex hull, the efficiency can be improved quickly. Therefore, the points of the planar point set are partitioned into m×n rectangular areas which are the same size. For the convenience of description, each rectangular area is called a cell. After such a partitioning , some cells contain points, and other cells may have no any point. The points belonged to a cell include not only those in the interior of the cell, but also those on the cell border. So, a point may be contained in several cells synchronously. But which cells are useful to the calculation of the convex hull? The Castle Theorem presented bellow reveals the rule.

B. Castle Theorem

In order to illuminate the Castle Theorem, we bring out some relative definitions and preliminary theorems first.

Definition 1: Suppose S is a planar point set, H(S) is the convex polygon that contains all the points of S, and for an arbitrary convex polygon H0(S) that contains the points of S, if there exists $H(S) \subseteq H0(S)$, then we call H(S) is the convex hull of the planar point set S. The vertexes of H(S) are called

convex hull points, the points that are not the vertexes of H(S) are called interior points.

Studies show, we can find the cells which are useful for calculating the convex hull through the following method. Suppose that there is an infinite rectangle, it can be partitioned into many cells, the size of each cell is entirely the same as that of the cell of the partitioned data area. The most exterior cells around the rectangle are transparent. We use this rectangle to detect the type of the cells: exterior of the castle, on the rampart or interior of the castle. We call it the Cell Type Detecting Template (Detecting Template, for short). It is showed on the left of Fig.1.

The rules for using the detecting template are as follows:

- The detecting template can be moved in parallel but can not be rotated, and it is parallel to the partitioned direction of the data area.
- The detecting template should be moved to the center of the data area from the outside as much as possible.
- The cells covered by the interior cells of the detecting template can not contain any point.

The concrete methods of using the detecting template can be seen in Definition 2.

Definition 2: Using the detecting template to detect the partitioned data area from each direction, the cells that can be covered by the interior cells of the detecting template are exterior of the castle; the cells only can be covered by the transparent cells of the detecting template is on the rampart; the rest cells that can not be covered by the detecting template are interior of the castle.

Obviously, the exterior cells of the castle do not contain any point.

Definition 3:The cells protruding outward on the rampart are called corner cells.

Theorem 1: Each corner cell of the castle contains one point at least.

Proof. Supposing the conclusion is false, that is to say the corner cell does not contain any point, because the exterior of the corner cell is exterior of the castle, if we detect by using the template, the corner cell will be covered by the interior cell of the template, so the cell will be the exterior cell, not on the rampart, even not the corner cell.□

Definition 4:The east, south, west, north, southeast, northeast, southwest and northwest of a cell are called neighbors of the cell.

Among the neighbors of different types, the number of the exterior cells exists certain regularity. There is none in exterior of the castle among the neighbors of an interior cell. Among the neighbors of a cell on the rampart, there is at least one neighbor cell in exterior of the castle (at the same time there is at least one neighbor cell in interior of the castle).

Theorem 2: Supposing S is a planar point set, P is a point in S, there are other four different points P1, P2, P3, P4, if P is in the quadrilateral formed by P1, P2, P3, P4, then P must be interior point.

Proof. Supposing the convex hull of planar point set is H(S), the quadrilateral Q is formed by P1,P2,P3,P4. Because P1, P2, P3, P4 must be convex hull point or interior point, Q must be in H(S) or inscribe H(S), and p is in Q, so P must be in H(S), that is to say P is an interior point of S.□

Theorem 3 (Castle Theorem): The convex hull points must be on the rampart.

Proof. All cells are classified into three types: exterior cells, rampart cells and interior cells. To prove it, we only have to prove that the points of the interior cells of castle must be interior points of the convex hull, because exterior cells of the castle do not contain any point.

The interior point p is classified into two types: ① on the boundary of interior or rampart cell; ②not on the boundary of interior or rampart cell . In the former, the point p is also on the rampart, so we only need to prove the latter.

In the latter, if c is an arbitrary cell containing point x, because the northeast, northwest, southeast, southwest of c must have corner cells, and every corner cell has a point at least. If we take an arbitrary point from four corner cells separately, and form a quadrilateral, then x must be in the quadrilateral. According to Theorem 2, we know that x is an interior point.□

Through Castle Theorem we can know, there is no point in exterior of the castle, the interior points of castle are all interior points. They are irrelevant for calculating the convex hull, only the points on the rampart are useful. If we can find the rampart in all cells quickly, we can get the candidate points for convex hull. Thus we can remove most of irrelevant points and improve the calculating efficiency. The Fast Rampart Searching Algorithm (FRSA for short) gives the answer for the problem.

III. ALGORITHM DESIGN AND REALIZATION

A. Main steps of algorithm

Main steps of calculating convex hull through FRSA is described as follows:

- Users input the partition number of rows and columns;
- Obtain the relevant statistical information of the point set in data table (maximum and minimum values of two dimensions);
- Calculate partition solution according to the partition number of rows and columns and the statistical information of the point set;
- Find the cells on the rampart through the FRSA algorithm;
- Fetch the point on the rampart from the table to form candidate set point;
- Remove the repetitive points of candidate set point (the repetitive points of the candidate point set are located at common border of two neighboring rampart cells, the number of them is less, so this step can be omitted);
- Calculate the convex hull from the candidate point set.

At last we can use any existing convex hull algorithm. Thereinto, the search of the rampart cells is a crucial step.

B. Fast Rampart Searching Algorithm，FRSA

When we search the cells on the rampart, we may carry on SQL queries to all cells in order to gain the points they contain, but this must increase unnecessary SQL queries, and reduce searching efficiency. This paper presents a Fast Rampart Searching Algorithm, which can search the cells from outside to inside layer by layer.

In order to search the rampart quickly, all cells are classified into five types :

0 --- Exterior of the castle;
1 --- Rampart, determine after searching;
2 --- Rampart, marking directly;
3 --- Interior of castle, judged;
4 --- Interior of castle, not judged;

The type of every cell is initialized as 4. Every kind of them has different background color. We can distinguish them during the searching visually.

Fast Rampart Searching Algorithm may be divided into two important steps:

- Searching the cell on the outmost layer;
- From the outer layer on, searching each interior layer in turn, until it is completed.

The searching method of the outmost layer is different from each interior layer, so we use two different steps. Additionally, the termination condition of the algorithm is very important too.

1)Searching the rampart cells of the outmost layer

The Outmost layer consists of the cells of four sides : the east, south, west and north cells. The algorithm searches the cells of each side separately, the concrete method is:

For each side, we scan each cell from two ends separately, and judge whether it contains at least a point, if not, its type is 0; otherwise we save the location (array index). The type of the first cell searching from two ends is 1, and the type of all cells between the two cells is 2.

2)Searching the rampart cells of each interior layer

We must rely on the status of the outer layer to search each interior layer of the rampart cells. The searching includes four sides too: the east, south, west and north. The pseudo-code of whole searching algorithm is as follows:

```
layer ←1;  // layer number searching
while(true){
    searching from the east cells of the current layer;
    searching from the south cells of the current layer;
    searching from the west cells of the current layer;
    searching from the north cells of the current layer;
    if(the searching is over ){
        break;  // exit the loop
    }
    layer ++;  // searching the next layer
}
```

Thereinto, the rampart cell searching algorithm of each side (for an example of the north)is as follows:

```
begin ← 0;
end   ← 0;
Handle each cell c from left to right as the follow step:
if(c is already rampart){
    begin ←current place;
    finished the searching along this direction;
}else{
    if(the outer cell of c is rampart){
        begin ←current place;
        finished the searching along this direction;
    }else{
        pointNumber←query the point number
        that this cell contains in data base;
        if(this cell contains point){
            begin ←current place;
            c.type ←1
            finished the searching along
            this direction;
        } else{
            c.type ←0
        }
    }
}
Handle each cell c from right to left as the follow step:
if(c is already rampart){
    end ←current place;
    finished the searching along this direction;
}else{
    if(the outer cell of c is rampart){
        end ←current place;
        finished the searching along this direction;
    }else{
    pointNumber←query the point number that this cell
    contains in data base;
        if(c contains point){
            end ←current place;
            c.type ←1
            finished the searching along this
            direction;
        }else{
            c.type ←0
        }
    }
}
```

Handle each cell c between “begin” and “end” individually as the follow step:

```
if(c.type==4){// not marked up
    if(the_outer_cell_of_c.type==0 or
    the_outer_cell_of_c.type==1){
        c.type←2;
    }else{
        if(the_outer_cell_of_c.type==2 or
            the_outer_cell_of_c.type==3){
            c.type←3;
        }
    }
}
```

Through the above algorithm, we found the rampart cells quickly. During the searching, we only need to query the database if it is necessary, so the amount of SQL queries is small in reality. The analysis shows that the amount of SQL queries is equal to the total amount of the exterior cells and the corner cells. Of course, if the data of the points is not

stored in database, we can distribute these points into every cells in O(n) time.

3)The criteria of finishing the searching

In this algorithm, the time for determining whether the searching is finished is after the searching of a layer. The criteria of finishing the search is that none is the exterior cell among all cells of current layer (constructing a rectangle).

C. Algorithm implementation

In order to verify the thought and algorithm, we finished the experiment under JBuilder 9. Fig.2 shows the experimental result of Fast Rampart Searching Algorithm. In the experiment, we partitioned the points into 30×30 cells, found the rampart which was composed by the cells with dark grey background color.

IV. ANALYSIS OF ALGORITHM EFFICIENCY

During calculating the candidate point sets using Fast Rampart Searching Algorithm, we need not spatial index and spatial search, nevertheless, we take advantage of the high performance of SQL queries, such that we can get the candidate point sets quickly. And its efficiency is mainly decided by the amount of SQL queries, which depends on the distribution of the data itself and the partition solution, that is the partition number of cells. In general, the finer the data area is partitioned, the more the amount of SQL queries, and the more the points that are got rid of.

The fast rampart searching algorithm uses less computer RAM. The algorithm only constructs an object array which includes m×n cells during searching the rampart. The queried data from the points table in the database is only the number of the points contained in every cell. In the process of calculating the convex hull through the candidate point set, we only need to store the candidate point set in the RAM.

The Excellent performance in space and time make the algorithm very fit for the preprocess to calculate convex hull from very large planar point set. It can get rid of most points that are useless for calculating convex hull, so the efficiency is improved obviously in the whole period.

V. CONCLUSION

Castle Theorem presented in the paper reveals a regularity of calculating the convex hull. We proved and verified it through the experiments. In the meantime, we explained the validity of the method. However, the algorithm can be used for calculating the convex hull of planar point set. The main research direction later is extending it to 3-dimension and higher dimensional space.

REFERENCES

[1] Shangmeng Wen, Feng Wang, Xiaomei Li, Xingming Zhou, "An O(Logn) Steps Parallel Algorithm for Finding the Convex Hull of Any Collection of N Points in the Plane", Chinese Journal of Computers , pp.828-831, September 1997.

[2] Jiyuan Liu, Xinsheng Wang, Dafang Zhuang , Wen Zhang, Wenyan Hu, "Application of Convex Hull in Identifying the Types of Urban Land Expansion", Acta Geographica Sinica ,pp. 885-892, June 2003 .

[3] Lihua Zhang Wenli Xu, "Convex Hull Based Point Pattern Matching Under Perspective Transformation", Acta Automatica Sinica, pp. 306-309, February 2002 .

[4] Rencan Peng, Jiayao Wang, Zhen Tian, Lixin Guo, Zipeng Chen, "A Research for Selecting Baseline Point of the Territorial Sea Based on Technique of the Convex Hull Construction",Acta Geodaetica Et Cartographic Sinica ,pp. 53-57, January 2005 .

[5] Peide Zhou, "Computational Geometry : Analysis and Design on the Algorithms", second edition, Tsinghua university press, Beijing, 2005.

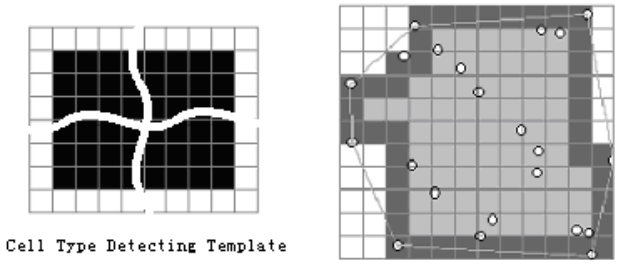

Figure 1. detecting the cell type using the Detecting Template

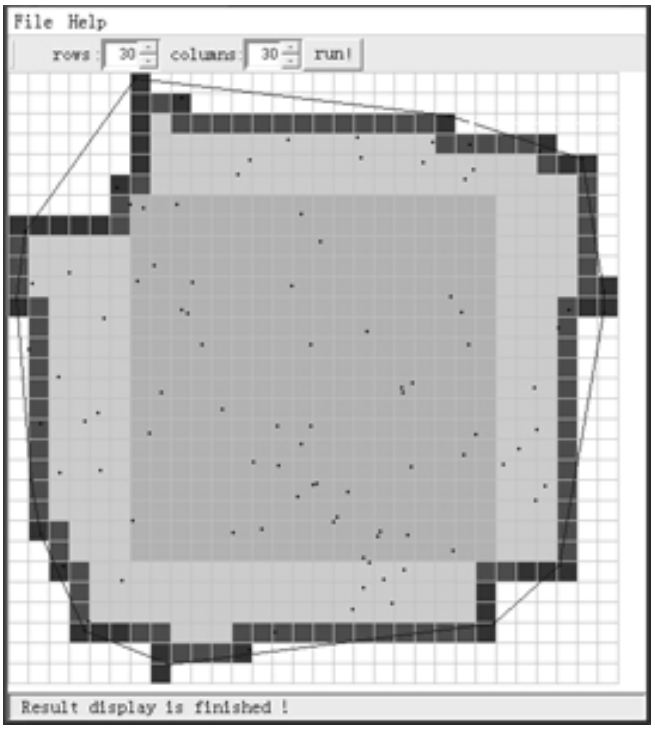

Figure 2. the experimental result of Fast Rampart Searching Algorithm