Indexing the Vector Data by Quadtree

Qing—huai Gao, Qing—yun Shi, Min—de Cheng

Information Science Center, Peking University,
100871, Beijing, P. R. China

Abstract

In the study of GIS/LIS, two kinds of data structure are studied: raster data structure and vector data structure. It is proved by experience and theory that the vector data has a higher compression rate and a well—organized logical structure (that is, an object is represented by its boundary polygons , and a polygon is a series of vectors v_1, v_2, ... ,v_N , where v_i and v_{i+1} have a same end point), and some basic image operations , such as scaling, rotating, calculating the area and perimeter , can be easily done on the vector data structure. But the spatial organization of the vector data is very loose , which leads to low efficiency when querying the objects near or at a given position. So, a good spatial index is needed to speed up the response of the system.

In this paper, it is suggested that the vector data can be organized by a quadtree—like index structure———— QO(quadtree of objects). The definition of QO , the algorithm for creating QO of a image, the algorithms of operation on QO are given. By analysing the querying effectiveness of QO, it is shown that the time consume of queries on the vector with QO — index is much more smaller than that on the "pure" vector data.

Keywords : GIS/LIS, Raster, Spatial , Data Base

§ 1 Introduction

The main task of the researcher on GIS is to speed up the implementation of variant queries on the data in some compressed form . In ordinary, the atom query of a GIS can be categorized into two classes : one called attribute—based query, and the other called position—based query. The sophisticated queries are all composed by a series of atom queries, which makes, in the view of users, the GIS manage the spatial data (image data) and the attribute data consistently and flexibly. Obviously , the position—based query has a more important position in the studies of GIS than the attribute— based query , and the speed of this kind query indicates the standard of a GIS. The class of the position—based query contains: to find the objects near or at a given position, to open a window on a image , to judge the geometric relation of two objects (for example , whether one object contains or intersects with the other object) , to query the objects which have some special geometric relation with a given object, etc.

In the study of GIS/LIS , two kinds of data structure are studied: raster data structure and vector data structure. It is proved by experience and theory that the vector data has a higher compression rate and a well—organized logical structure (that is, an object is represented by its boundary polygons , and a polygon is a series of vectors v_1, v_2, ... ,v_N , where v_i and v_{i+1} have a same end point), and some basic image operations , such as scaling, rotating, calculating the area and perimeter , can be easily done on the vector data structure. But the spatial organization of the vector data is very loose , which leads to low efficiency when querying the objects near or at a given position. So, a good spatial index is needed to speed up the response of the system.

In this paper, it is suggested that the vector data can be organized by a quadtree—like index structure ———— QO (quadtree of objects). In a node of QO , a list of objects is stored. A object is stored in the node which correspondences to the smallest quad block that covers this object.

In the second section , the definition of QO and the algorithm for creating QO of a image , the algorithms of operation on QO are given. In the third section , the time consume of queries in the class of position—based query are analysed . In the last section a short conclusion is given.

§ 2 The definition and operation on QO

Organizing objects according their 2D position is the main idea of QO.

2. 1 The definition of QO

Suppose that the image f is a function on region R with size $2^n * 2^n$. Let set $SO = \{o_1, o_2, \ldots, o_N\}$ be the objects on the image , and $R(o_i)$ represents the region correspondences to o_i .

A QO is a tree whose every node has 4 or no sons, that is , a node of QO is a structure shown below:

```
structure node {
    structure node *Son1, *Son2, *Son3, *Son4;
    Set—of—Objects S ;
}
```

Now , a recursive algorithm of creating QO is given below as the definition of QO.

Algorithm : $CreatQO(R, SO)$

Procedure :

step 1: Get a node r as the root of the QO, set every item of r null.

step 2: Let R_i , where $i = 1,2,3,4$, is the four quad block of R ; For $\forall\ o \in SO$,if $R(o) \subseteq R$ and $R(o) \not\subseteq R_i$, $i = 1,2,3,4$, then

$$r.S = r.S \cup \{o\};$$

step 3: Let $SO_i = \{o \mid o \in SO - r.S, R(o) \subseteq R_i\}$, $i = 1,2,3,4$; If SO_i is not empty , then create $QO_i = CreatQO()$ by recursion, and let $r.Son_i$ points to QO_i, $i = 1,2,3,4$.

2.2 The operation on QO

The inserting, deleting and searching are the main basic operations on QO . We give the inserting algorithm on QO below.

```
INSERTQO( r,o,R )
Begin
    if r =NULL
    then get a node r as the root of QO, and
    set all the items in r null.
    if R(o) ⊆ R
    then begin
        r.S = r.S ∪ {o} ;return;
    end
    Let Ri is the ith quad block of R ,
    if R(o) ⊆ Ri
    then begin
        if r.Soni =NULL
        then get a node ,and let r.Soni
     points to it.
        INSERTQO( r.Soni,o,Ri );
    end ;
End
```

The deleting and searching algorithm are similar to the inserting algorithm, so no details are given here.

Using the inserting algorithm, an algorithm of creating QO with time consumption $O(N)$ can be given.

2.3 QO and position—based queries

(1) To find the objects at a given position (x,y) : only check the objects stored at the nodes whose corresponding quad blocks contain the point (x,y).

(2) To open a window on a image : if the set of nodes , whose corresponding quad blocks intersect with the window, is $\{nd_1, nd_2, \ldots, nd_m\}$, then only check the objects stored at all the ancestor nodes of nd_i, $i = 1, \ldots, m$, and the objects stored at the subtree whose root is nd_i.

(3) To find all the objects contained in or intersecting with a given region: At first , look up all the nodes whose corresponding quad blocks intersect with the region , then check the objects on these nodes.

2.4 The organization of objects stored at a node

Because the queries on QO are transferred to search on some nodes on the tree, so it is necessary to organize the objects stored at a node in a suitable way . A schema is to store the objects by a sorted balanced binary tree with the size of objects as the key. Using this structure can further confine the searching scope and leads to a higher effectiveness.

§ 3 The Querying Effectiveness of QO

Using QO, the searching in queries are confined at one node or some nodes. So, in this section, we will discuss the effectiveness by analysing the number of objects stored at a node or a series of nodes.

To simplify the discuss, we suppose the image corresponding to a region of unit square $[0,1]^2$, similarly , suppose that the total number of objects on the image is 1.

The diameter (or size) $d(R)$ of a region R is defined as $d(R): max\{d(x_1,x_2) \mid x_1, x_2 \in R\}$.

Suppose the probability density of $d(R)$ is $p(x)$, where $x \in [0, \sqrt{2}]$.

3.1 The expected number p_m of objects stored at a node

Suppose that node nd corresponding to a quad block B with size , the expected number of objects stored at nd is p_m . We divide p_m into two parts: one corresponding to the objects whose size is larger than $1/2^{m+1}$, the other part corresponding to the else, and we use p_{m1}, p_{m2} to represent them, so , there is $p_m = p_{m1} + p_{m2}$.

Because

$$p_{m1} \leqslant \frac{1}{2^{2m}} \int_{1/2^{m+1}}^{\sqrt{2}/2^m} p(x)dx$$

$$p_{m2} \leqslant \int_0^{1/2^{m+1}} p'(x)p(x)dx$$

where $p'(x)$ is the probability of the objects with size x intersecting with the lines which divide B into four equaling sub blocks.

Under a model with equaling probability , it is easy to deduce that $p'(x) \leqslant \frac{4x}{2^m} - x^2$, So we have

$$p_m \leqslant$$

$$\frac{1}{2^{2m}}\int_{1/2^{m+1}}^{\sqrt{2}/2^m} p(x)dx +$$

$$\int_0^{1/2^{m+1}} (\frac{4x}{2^m} - x^2)p(x)dx$$

3.2 The expected number $P(B)$ of objects which intersect with a quad block B.

Suppose a quad block B with size $1/2^m * 1/2^m$ correspondences the node nd_m. The objects which intersect with B are stored at the sub tree of QO whose root is nd_m, or at one of the ancestors of nd_m. Suppose that the ancestors of nd_m are $nd_1, nd_2, \ldots, nd_{m-1}$, where nd_0 is the root of QO, and p_i is the expected number of objects stored at nd_i, and q_m is the expected number of objects stored at the subtree whose root is nd_m, then

$$P(B) \leqslant \sum_{i=0}^{m-1} p_i + q_m$$

To estimate $\sum_{i=0}^{m-1} p_i + q_m$, we estimate p_{i3}, which is the expected number of objects stored at the subtree whose root is nd_i but not at the tree whose root is nd_{i+1}, and we have:

$$p_{i3} \geqslant 3\int_0^{1/2^{i+1}} (1/2^{i+1} - x)^2 p(x)dx$$

then

$$P(B)$$

$$\leqslant \sum_{i=0}^{m-1} p_i + q_m$$

$$= 1 - \sum_{i=0}^{m-1} p_{i3}$$

$$\leqslant 1 - \sum_{i=0}^{m-1} 3\int_0^{1/2^{i+1}} (1/2^{i+1} - x)^2 p(x)dx$$

3.3 A example of p(x)

Now using a example of $p(x)$, we further estimate p_m and $P(B)$, to show the effectiveness of QO.

Let $p(x) = e^{-1/\sigma x}/(\sigma e^{-1/\sqrt{2}\sigma} x^2)$, its shape shown on Figure 1.

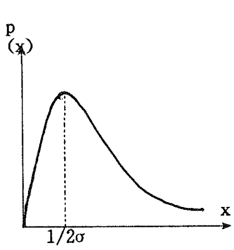

Fig. 1 The shape of p(x)

The reason of letting p(x) be this special function is listed below: (1) the number of objects with size x containing in the unit square potition with $1/x^2$; (2) for a given class of images, the objects which is two smaller is always noise, so we set a decreasing factor $e^{-\frac{1}{\sigma x}}$ 。 $p(x)$ reach its maximum value at point $x=\frac{1}{2\sigma}$, so σ shows the maximum feature of the probability density function of the size of objects.

1. The estimate of $P(B)$

We obtain the estimate below:

$$P(B)$$

$$\leqslant \frac{1}{4^m}$$

$$+ \frac{3}{\sigma}(1 - \frac{1}{2^m})$$

$$+ \frac{6}{(4-\sqrt{2})\sqrt{\sigma}}(1 - (\frac{\sqrt{2}}{2})^{3m})$$

2. The estimate of p_m For p_m, we obtain:

$$p_m \leqslant \frac{1}{2^m}(\frac{(4-\sqrt{2})}{2\sigma} +$$

$$\frac{2}{(\sqrt{2})^m \sqrt{\sigma}})$$

§ 4 Conclusion

The vector data has a well—organized logical structure, but has no natural spatial index. In this paper, a spatial index for vector data, called QO, is presented. Analysing the querying effectiveness on the vector data with QO structure, it is shown that the time consume of queries in the position—based class is much smaller than that on "pure" vector data.

This structure also can be used in other systems, such as CAD/CAM systems or vision systems.

References

[1] A. Rosenfeld and A. Kak, Digital Picture Processing, 2nd Ed., Academic Press, New York, 1982.

[2] S. K. Chang, Principles of Pictorial Information System Design, Prentice—Hall, Inc., 1989.

[3] M. J. B. Duff, "Intermediate—level Image Processing", London Academic Press, 1986.

[4] H. Samet, "The Quadtree and Related Hierarchical Data Structures", Computing Survey, Vol. 16, No. 2, 1984, 187—260.

[5] E. Kawaguchi and T. Endo, et al, "Depth—first Expression Viewed from Digital Picture Processing", IEEE Trans. Vol. PAMI—5, 1983, pp373—384.

[6] Q. Y. Shi, "Image Processing Operations Using CD Representations", Proc. of 8th ICPR, 1986, pp320—322.

[7] S. X. Li and M. H. Leow, "The Quadcode and its Applications", Proc of 7th ICPR 1984, pp227—229.

[8] H. Samet and M. Tamminen, Computing Geometric Properties of Images Represented by Linear Quadtree",

IEEE Trans. on PAMI, Vol. PAMI—7, No2, March 1985.
[9] S. K. Chang and T. Kunii, "Pictorial Data—Base Systems", Computer, Vol. 14, No. 11, 1981, pp13—22.