第 30 卷 第 2 期
2001 年 5 月

测 绘 学 报
ACTA GEODAETICA et CARTOGRAPHICA SINICA

Vol. 30, No. 2
May., 2001

文章编号: 1001-1595(2001)02-0156-06 中图分类号: P208 文献标识码: A

一种面向地理信息系统的空间索引方法

史文中[1], 郭 薇[1, 2], 彭奕彰[1]
(1. 香港理工大学 土地测量与地理资讯系, 香港 九龙; 2. 武汉大学 计算机科学学院, 湖北 武汉 430072)

A Spatial Indexing Method for GIS

SHI Weng-zhong[1], GUO Wei[1, 2], PANG Yick-cheung[1]
(1. *Department of Land-Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong, China*; 2. *Institute of Computer Science, Wuhan University, Wuhan* 430072, *China*)

Abstract: It has been recognized in the past that the traditional Database Management Systems cannot handle efficiently spatial data in mutil-dimensional space. This paper focuses on deriving efficient access methods for spatial objects in Geographical Information Systems (GIS). First, we discuss the classification of existing spatial indexing methods and point out the problems of them. Second, the approximate representation of spatial objects is given. Third, the structure and the algorithm of CP-tree, which based on the convex polyhedra, are proposed in detail. Finally, we compare search performance between CP-tree and R-tree. It is found that CP-tree requires a few CPU time and disk accesses for region searching than the R-tree. Thus, it enhances the spatial query performance remarkably.

Key words: GIS; spatial indexing; R-tree; R^+-tree; CP-tree; convex polyhedra

摘 要: 空间检索技术是有效地管理和操纵空间数据的一种必要手段。本文分析了目前常用的空间检索方法在地理信息系统应用上的局限性, 提出了一种基于凸多边形的空间索引技术。本文首先介绍了目前常用的几类空间索引技术及其特点, 接着讨论了面向地理信息的空间索引技术面临的基本问题, 提出了基于凸多边形的空间索引结构——CP-树, 并侧重分析了其空间运算算法及时空效率, 最后指出了空间索引技术所存在的问题及下一步的研究方向。

关键词: 地理信息系统; 空间索引; R-树; R^+-树; CP-树; 凸多边形

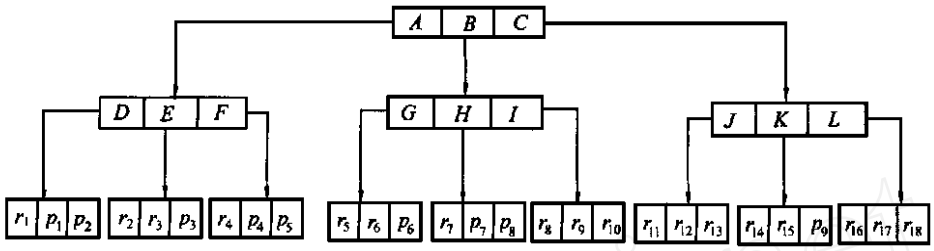
1 前 言

地理信息系统(Geographical Information System, 简称 GIS)的主要任务之一是有效地检索空间数据及快速响应不同用户的在线查询。因此, 建立一种面向地理信息系统的空间索引方法, 以有效地实现空间数据的分析和处理, 具有十分重要的意义。

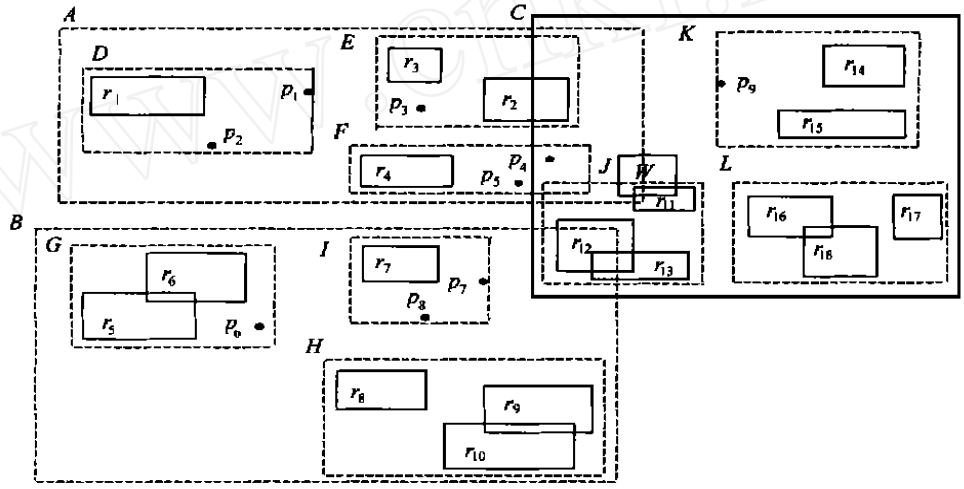
R-树是目前应用最为广泛的一种空间索引结构[1~4]。空间目标采用最小约束矩形(Minimum Bounding Rectangles, 简称 MBR)法来加以近似表达。R-树具有较高的空间效率, 它可以保证其

收稿日期: 1999-07-20; **修回日期**: 2001-01-04
基金项目: 香港研究资助局基金项目(PolyU 5093)
作者简介: 史文中(1963-), 男, 香港理工大学副教授, 博士, 主要从事地理信息系统研究。

空间利用率在 50% 以上[1]。然而, R-树是通过其动态插入算法来建立的, 其结构不可避免地会导致约束矩形间相互重叠区域及空白空间的出现, 当查找与给定的查询窗口相交得所有空间目标时, 空间搜索算法是从根结点开始, 向下搜索相应的子树。算法递归遍历所有约束矩形与查询窗口相交的子树, 当到达叶结点时, 边界矩形中的元素被取出并测试其是否与查询矩形相交, 所有与查询窗口相交的叶结点即为要查找的空间目标。例如, 对图 1 所示的 R-树, 给定查询窗口 W, 由于 W 既落在结点 A 所对应的约束矩形中, 又落在结点 C 所对应的约束矩形中, 因此, 虽然只有结点 C 中的目标 r_{11} 与查询窗口 W 相交, 但结点 A 和结点 C 对应的子树都必须进行搜索, 从而导致查询效率的降低。研究表明, 对于非点状目标而言, 很难使得 R-树中位于同一层的 2 个或多个结点对应的最小约束矩形之间的公共区域为零。在 R-树的动态分裂中, 也很难控制其重叠区域, 且重叠区域随着空间目标维数的增大而急速增大[5]。因此, R-树的查询效率会因重叠区域的增大而大大减弱, 在最坏情况下, 其时间复杂度甚至会由对数搜索退化成线性搜索。



(a) R-树



(b) 查询与窗口 W 相交的目标

图 1　R-树及其区域查询

Fig. 1　The structure of R-tree and query spatial objects intersected with window W

为了克服 R-树中最小约束矩形相互重叠的缺陷, R^+-树的概念被提出来了[6]。R^+-树避免了最小约束矩形相互重叠导致的多路径查询问题, 提高了空间检索效率。然而, 当 R^+-树克服了 R-树中最小约束矩形相互重叠问题的同时, 它也存在着一些缺陷, 如对某个最小约束矩形的划分, 可能会引起相关子树上其他结点也需要重新划分, 向下分裂操作可能使得已经划分好了的结点被重新划分, 空间目标在 R^+-树的叶结点中被重复标记, 完成删除运算后, 必须对 R^+-树进行重建等。

从以上分析中可以看出, 空间索引技术还存在着许多问题有待进一步探讨。要提高索引结构的空间搜索能力, 一个必要的步骤是既要使得搜索树中, 非叶结点所对应的最小约束矩形之间相互重叠的区域尽可能小, 又要尽量避免由于目标复制带来的空间利用率不高, 数据管理较为困难

等缺陷。基于以上观点,本文提出了一种基于凸多边形的空间目标近似表达方法。在此基础上,设计了一种面向地理信息系统的空间索引结构——CP-树。本文首先论述了CP-树的基本结构及特点,然后给出了详细的空间操作算法,并就该结构与其他结构的时间和空间效率给出了实验性评估,最后,给出了下一步的研究方向。

2 基于凸多边形的空间索引结构

2.1 基于凸多边形的空间目标近似表达方法

一般来说,GIS 所处理的空间目标具有不规则的形状,若基于它们的精确位置和扩展来实现某些空间操作(如相邻,包含等),其计算量会非常庞大。因此,一些近似的方法,如最小约束矩形法等,常常被用来表达具有不规则形状的空间目标。空间目标近似表达方法可以简化某些空间查询过程,避免一些不必要的计算,从而可以有效地提高查询效率。

本文采用凸多边形(convex polyhedra)来近似表达空间目标。凸多边形是 d 维欧氏空间 R^d 中一个非空、有限的凸集。对 R^d 中的一个集合,如果这个集合中的任意 2 个点,连接这 2 个点的线段被完全包含在该集合中,则该集合被称作凸集。研究表明,对于任意一个空间目标而言,这个目标对应的凸多边形必然包含在该目标对应的最小约束矩形中[7]。因此,凸多边形能够更精确地定义空间目标的位置,减少不同空间目标之间的重叠区域,从而可以更有效地实现空间查询。

2.2 CP-树的索引结构

CP-树是一种基于凸多边形的空间索引结构,它是B-树在多维空间中的一般化形式,因此,与B-树类似,它也是一种平衡树,当插入或删除空间目标时,需要对结点进行分裂和合并。CP-树由叶结点和非叶结点 2 个部分所构成。

叶结点 L 具有如下结构:

$$L:(E_1,\cdots,E_n)\quad(n\leq ML)$$

$$E_i:(\text{object-identifier},\ \text{CPOLY})$$

其中,ML 是叶结点 L 中可以包含的最大目标数目;E_i 是包含在叶结点 L 中的空间目标,它包含有目标标识号(object-identifier)及包围该目标的 k 维凸多边形(CPOLY)。

非叶结点 N 具有如下结构:

$$N:(C_1,\cdots,C_n)\quad(n\leq MN)$$

$$C_i:(\text{child-pointer},\ \text{CPOLY})$$

其中,MN 是非叶结点 N 可以包含的最大目标数目;C_i 包含有指向R-树中较低一级结点的孩子指针(child-pointer)及一个凸多边形(CPOLY),这个凸多边形包围了该非叶结点所有子树结点对应的凸多边形。图 2 给出了一个与图 1 所示的R-树具有相同树结构的CP-树的平面表示。

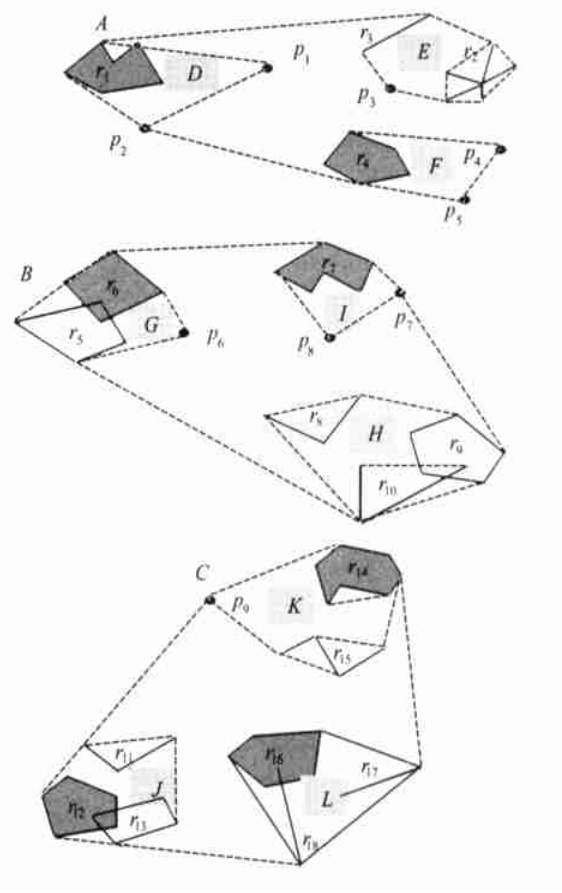


图 2 CP-树的平面表示

Fig. 2 The planar representation of an CP-tree

2.3 搜索运算

CP-树的搜索算法与R-树类似,空间搜索算法是从根结点开始,向下搜索相应的子树。算法递归遍历所有凸多边形与查询窗口相交的子树,当到达叶结点时,边界矩形中的元素被取出并测试其是否与查询矩形相交,所有与查询窗口相交的叶结点即为要查找的空间目标。其算法如下:

Algorithm Search (N, W);

{在以 N 为根结点的CP-树中,搜索与查询窗口 W 相重叠的所有空间目标}

Begin

If N < > NIL do

If N: $(C_1, \cdots, C_n)$ is a non-leaf node Then
For i: = 1 to n do
[check C_i: (child-pointer, CPOLY);
If CPOLY overlaps with W Then Search (C_i. child-pointer, W);]
If N: $(E_1, \cdots, E_n)$ is a leaf node Then
For i: = 1 to n do
[check E_i: (object-identifier, CPOLY);
If CPOLY overlaps with W Then return (object-identifier)]
End.

2.4 生成运算

CP-树是一个动态结构。CP-树的生成是从空树开始, 逐个插入空间目标而得。对于一个M 阶的 CP-树而言, 它的每个结点至少含有$(M/2)$个元素, 因此, 当在 CP-树中插入一个空间目标时, 其插入运算包括 2 个步骤: 首先从根结点出发, 向下查找一个非叶结点, 使得插入新目标后, 该非叶结点所对应的凸多边形面积增加得最小。当这个结点包含的元素个数不超过$(M-1)$时, 直接将新结点插入至该结点中, 否则, 就调用分裂结点的算法, 将这$(M+1)$个目标划分成 2 个结点, 再向上插入这 2 个结点。其算法如下:

Algorithm Insert(N, P);
{在以N 为根结点的 CP-树中, 插入一个空间目标 P: (object-identifier, CPOLY)};
Begin
If N: $(E_1, \cdots, E_n)$ is a leaf node Then
[If $n < ML$ Then [P --> N: $(E_1, \cdots, E_n)$; n: = $n+1$)]
Else SplitNode]
Else If N: $(C_1, \cdots, C_n)$ is a non-leaf node Then
[Choose (C_i); {Choose the object C_i in N whose convex polyhedra needs least area enlargement to include the new convex polyhedra CPOLY}
Insert (C_i. child-pointer, P)]
End;
Algorithm SplitNode;
{将$ML+1$ 个目标 $E_1, E_2, \cdots, E_{ML}, E_{ML+1}$ 划分成 2 个集合, 分别放在 2 个不同的结点中}
Begin
For i: = 1 to ML do
For j: = $i+1$ to ML do
[R: = Combine (E_i. CPOLY, E_j. CPOLY); {R is the convex polyhedra composed by E_i and E_j}
d: = area (R) - area (E_i. CPOLY) - area (E_j. CPOLY);
If d = Max (area) then [G_1: = i; G_2: = j]]; {Choose the pair of entries E_i and E_j with the largest d; E_i and E_j to be the first entries of two groups G_1 and G_2}
Repeat
If E_i not in G_1 or G_2 then
For i: = 1 to $ML+1$ do
[d_1: = area (Combine (G_1. CPOLY, E_i. CPOLY));
d_2: = area (Combine (G_2. CPOLY, E_i. CPOLY));
d: = $d_1 - d_2$;
If d > Max (d) Then [Max (d): = d; G: = i]]; {Choose the entry E_i not yet in a groups with the maximum difference between d_1 and d_2}
If $d_1 > d_2$ Then E_i--> G_2 Else E_i--> G_1;
Until (All entries are distributed or one of the two groups has $M-m+1$ entries);
If entries remain, assign them to the other group such that it has the minimum number m;
End.

空间目标所对应的凸多边形可由该目标的顶点集合来构造, 本文所采用的凸多边形生成算法是由 Eddy 等人于 1977 年左右提出的快速凸多边形生成算法, 算法详细描述见参考文献[8]。

3 实验评估

为了比较 R-树和 CP-树的搜索行为, 本文设计了以下几个指标来测试 CP-树与 R-树的性能: 覆盖范围(coverage) 平均值、重叠区域 (overlap) 平均值、生成运算时所需的 CPU 时间及磁盘读写次数、区域查询时所需的 CPU 时间及磁盘读写次数。

R-树和 CP-树中某一层的覆盖范围分别指的是树中与该层结点对应的所有最小约束矩形或凸多边形的面积之和; 某一层的重叠区域指的是该

层中两个或多个结点对应的最小约束矩形或凸多边形之间的公共区域。显然,一个高效的R-树或CP-树要求其覆盖范围和重叠区域都尽可能小。因为较小的覆盖范围可以使得空白空间尽可能小,较小的重叠区域可以有效地提高查询效率。

设R-树或CP-树的深度为H,第i层中含有N个结点$\{P_{i1}, P_{i2}, \cdots, P_{iN}\}$,其中任意一个结点$P_{ij}$含有$M$个元素$\{P_{ij,1}, P_{ij,2}, \cdots, P_{ij,M}\}$,结点$P_{ij,K}$所对应的最小约束矩形的面积为$R_{area(P_{ij,K})}$,凸多边形的面积为$CP_{area(P_{ij,K})}$,则R-树中第$i$层的覆盖范围($R_{coverage(i)}$)及重叠区域($R_{overlap(i)}$)为

$$R_{coverage(i)} = \sum_{j=1}^{N}\left(\frac{\sum_{k=1}^{N} R_{area(P_{ij,k})}}{R_{area(P_{ij})}}\right)\Big/N$$

$$R_{overlap(i)} = \frac{\left|\{p \mid p \in \bigcap_{j=1}^{N} P_{ij}\}\right|}{\left|\{p \mid p \in \bigcup_{j=1}^{N} P_{ij}\}\right|}$$

其中,$|A|$表示集合A中元素的数目。

R-树覆盖范围及重叠区域的平均值为

$$R_{average\ coverage} = \sum_{i=1}^{H} R_{coverage(i)}\Big/H$$

$$R_{average\ overlap} = \sum_{i=1}^{H} R_{overlap(i)}\Big/H$$

CP-树中第i层的覆盖范围($CP_{coverage(i)}$)及重叠区域($CP_{overlap(i)}$)为

$$CP_{coverage(i)} = \sum_{j=1}^{N}\left(\frac{\sum_{k=1}^{M} CP_{area(P_{ij,k})}}{CP_{area(P_{ij})}}\right)\Big/N$$

$$CP_{overlap(i)} = \frac{\left|\{p \mid p \in \bigcap_{j=1}^{N} P_{ij}\}\right|}{\left|\{p \mid p \in \bigcup_{j=1}^{N} P_{ij}\}\right|}$$

CP-树覆盖范围及重叠区域的平均值为

$$CP_{average\ coverage} = \sum_{i=1}^{H} CP_{coverage(i)}\Big/H$$

$$CP_{average\ overlap} = \sum_{i=1}^{H} CP_{overlap(i)}\Big/H$$

为了评估CP-树的性能,本实验选用了处理器为Pentium II 333MHz的PC机,其显示器分辨率为1 280×1 024。本实验选用了2组实验数据,一组为均匀分布的数据集合,该集合包括一系列随机产生的空间目标(见图3),另一组数据为现实世界的空间目标(见图4)。

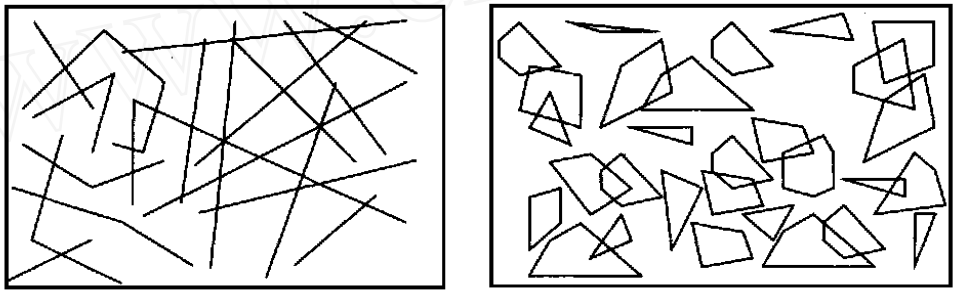


图3 一组随机生成的凸多边形

Fig. 3 A set of convex polygon generated randomly

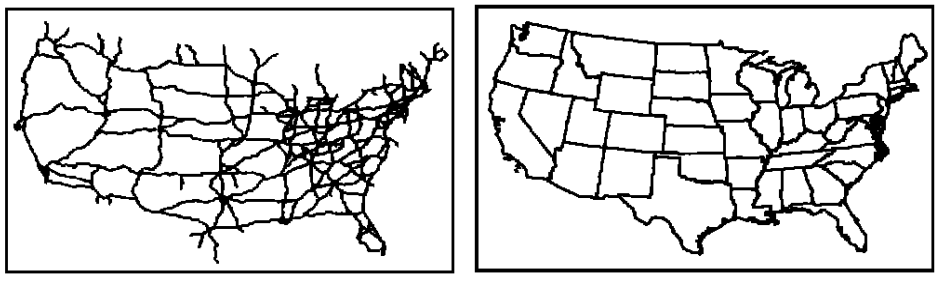


图4 现实世界目标

Fig. 4 A set of natural objects

本实验选用了结点数目分别为5 000, 10 000, 20 000, 30 000的四组空间目标集合。对于随机产生的空间目标及现实世界真实目标,其R-树和CP-树的测试指标值分别如图5和6所示。

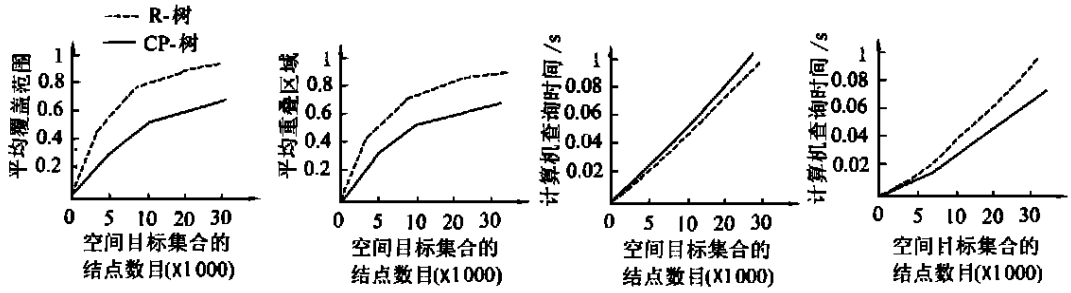


图 5　随机目标 R-树和 CP-树的性能比较

Fig. 5　Performance of R-tree and CP-tree for randomly generated objects

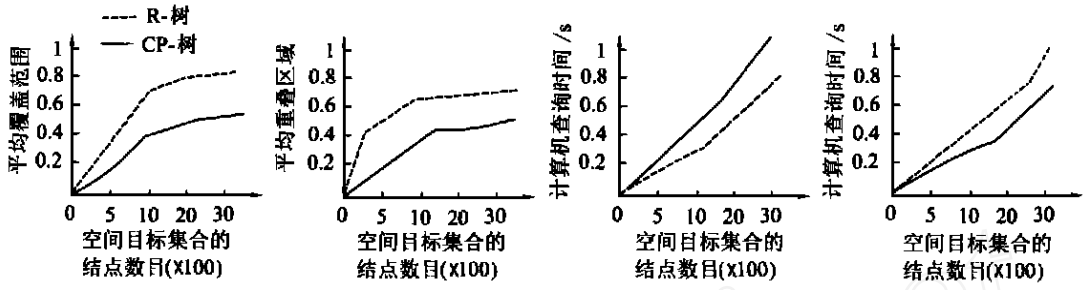


图 6　现实世界目标 R-树和 CP-树的性能比较

Fig. 6　Performance of R-tree and CP-tree for natural objects

4　结　论

空间索引技术在计算机辅助设计与制造(CAD/CAM)、地理信息系统(GIS)、图像处理(image processing)、虚拟现实语言设计(VRML)、数字地球(digital earth)等诸多领域均具有十分重要的研究意义,传统的文件检索方法难于满足空间搜索的要求。本文分析 GIS 中的几类空间目标,以一种最为常用的空间索引方法——R-树为基础,针对其存在的问题,提出了一种基于凸多边形的空间索引结构——CP-树,并侧重分析了其结构特点及在不同情况下的空间搜索效率及空间存贮效率。研究及实验表明,CP-树的覆盖范围及重叠区域较 R-树明显减小,因此,它有效地减少了 R-树中多路径查询的情形,从而明显地提高了其查询性能。

空间索引技术还存在着很多问题有待进一步解决。相关的研究结果,如扩展 CP-树至高维空间的分析、计算;改进凸多边形的生成算法,以提高算法效率等,我们另文论述。

参考文献:

[1] GUTTMAN A. R-trees: A Dynamic Index Structure for Spatial Searching [A]. Proceeding of ACM SIGMOD [C]. Boston: ACM Press, 1984. 47-57.

[2] GREENE D. An Implementation and Performance Analysis of Spatial Data Access Methods [A]. Proceeding of the 5th International Conference on Data Engineering [C]. Los Angeles: IEEE, 1989. 606-615.

[3] OOI B C. Efficient Query Processing in Geographical Information Systems [M]. New York: Springer-Verlag, 1990.

[4] BERTINO E, *et al*. Indexing Techniques for Advanced Database Systems [M]. Boston: Kluwer Academic Publishers, 1997.

[5] BERCHTOLD S. The X-tree: An Index Structure for High-Dimensional Data [A]. Proceeding of the 22th International Conference on Very Large Data Bases [C]. Brighton: Morgan Kaufamann, 1996. 28-39.

[6] SELLIS T, *et al*. The R^+-tree: A Dynamic Index for Mutil-Dimensional Objects [A]. Proceeding of the 13th International Conference on Very Large Data Bases [C]. Brighton: Morgan Kaufamann, 1987. 507-518.

[7] JAGADISH H V. Spatial Search with Polyhedra [A]. Proceeding of the 6th International Conference on Data Engineering [C]. Los Angeles: IEEE, 1990. 311-319.

[8] PREPARATA F P, SHAMOS M I. Computational Geometry: An Introduction [M]. New York: Springer-Verlag, 1985.