文章编号:1674-2974(2020)04-0099-11 DOI:10.16339/j.cnki.hdxbzkb.2020.04.013

一种存储复杂多边形包含关系的四叉树索引

汪红松†,周晓光
(中南大学 地球科学与信息物理学院,湖南 长沙,410083)

摘 要:地表覆盖/土地利用矢量数据中存在大量包含成千上万个空洞(甚至嵌套空洞)的复杂多边形,现有空间数据索引没有表达复杂多边形及其空洞之间的包含关系,导致空间数据冲突检测与更新等处理存在计算量大、效率低等问题.针对此问题,提出了一种存储多边形包含关系的四叉树索引方法.该方法根据结点中的多边形与四叉树相应象限中轴线相交的方式将多边形对象分为 5 种类型,即仅与 X 正轴相交、仅与 X 负轴相交、仅与 Y 正轴相交、仅与 Y 负轴相交以及与 XY 轴都相交,并将这些多边形对象分别存储在相应层次索引结点中的 5 个子列表(桶)中,然后在结点多边形对象中存储多边形之间的父子包含关系.最后设计并实现了该索引及相应的查询、插入、删除等算法,并用实际地表覆盖数据验证了本文方法的有效性.实验结果表明,采用本文索引方法的复杂地表覆盖矢量数据增量更新效率数倍于现有四叉树索引方法,且随着数据量的增加效率提高更明显.

关键词:空间索引;复杂多边形;包含关系;四叉树;空间数据管理
中图分类号:P208 **文献标志码**:A

A Quadtree Spatial Index Method with Inclusion Relations for Complex Polygons

WANG Hongsong†,ZHOU Xiaoguang
(School of Geosciences and Info-Physics,Central South University,Changsha 410083,China)

Abstract:There are a large number of complex polygons containing thousands of holes (or even nested holes) in the land cover/land use vector data, and the existing spatial data indexing method has failed to indicate the inclusion relationship between complex polygons and their holes, resulting in computationally heavy and inefficient processing such as spatial data conflict detection and updating. In order to solve this problem, an improved quadtree spatial index method with inclusion relations of the complex polygons is presented in this paper. The method classifies the polygons in the nodes into five types according to the way they intersect the axes in the corresponding quadrant of the quadtree, i.e., intersect only the X positive axis, intersect only the X negative axis, intersect only the Y positive axis, intersect only the Y negative axis, and intersect both X and Y axes, and stores each of these polygons in five sublists (buckets)

* **收稿日期**:2019-04-11
基金项目:国家自然科学基金资助项目(41371366),National Natural Science Foundation of China(41371366)
作者简介:汪红松(1975—),男,安徽怀宁人,中南大学博士研究生
† 通讯联系人,E-mail:whs2020@126.com

in the corresponding hierarchical index nodes, and then stores the parent-child inclusion relationship between the polygons in the node polygon objects. The authors developed the spatial index structure with inclusion relations and the algorithms of the corresponding operations (e.g., insert, delete and query) for the complex polygons. The effectiveness of the approach in this paper is verified by an experiment of land cover data incremental updating, experimental results show that the time efficiency of the incremental updating is increased about several times using the proposed index method than that of the traditional quadtree index, and the improvement in efficiency is more significant with increasing data volume.

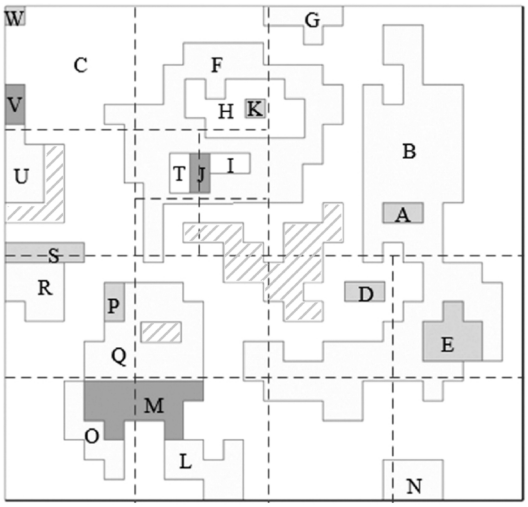
Key words: spatial index; complex polygon; inclusion relation; quadtrees; spatial data management

随着全球 30 m 地表覆盖地图 GlobeLand30[1-2]和全球 10 m 地表覆盖数据 FROM-GLC10[3]的完成与发布，全球地表覆盖数据已成为联合国等国际组织开展全球变化与可持续发展等重大科学研究的基础数据. 全球地表覆盖数据的验证、服务与持续更新成为本领域的研究热点[4-8].在全球地表覆盖矢量数据更新方面,周晓光等[7]提出了一种基于二维交细分拓扑关系的地表覆盖/土地利用数据增量更新方法,但由于地表覆盖矢量数据中存在大量包含成千上万个空洞(甚至嵌套空洞)的复杂多边形,目前空间数据模型中没有表达复杂多边形及其空洞之间的包含关系，导致在计算增量多边形与已存在的复杂多边形间二维交时存在计算量大、效率低等问题.

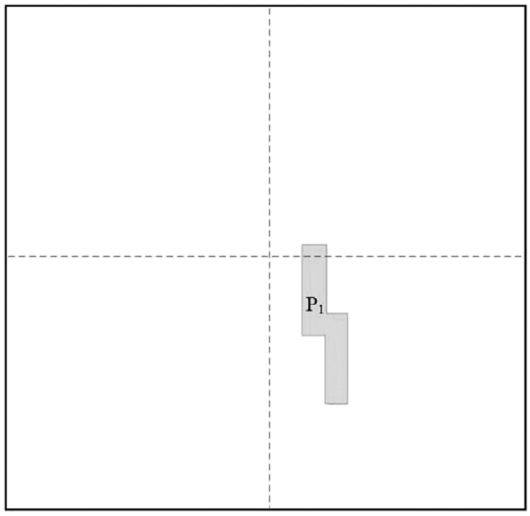
GIS 中空间数据处理一般包括过滤和精化计算两个步骤，空间数据索引用来过滤掉大部分无关的目标，使得精化计算仅在少数密切相关目标间进行的效率提升的特殊空间数据结构. 目前,空间数据索引方法包括传统矢量数据索引的四叉树[9-10]、R 树[11-12]、R+树[13]、R* 树[14-15]、网格索引[16]、Hilbert R 树[17]等和轨迹数据索引 Geohash-Trees[18]等. 上述传统矢量数据索引方法均采用目标的最小外接矩形(MBR) 减小索引结构的存储量并提高过滤效率. 但是对于复杂多边形，现有索引方法仅存储了其外边界的 MBR,不能表达复杂多边形及其空洞间的包含关系,在空间数据冲突检测与更新处理中,无关空洞不能通过索引而过滤掉,大量空洞需参与精化计算,是导致计算量大、效率低等问题的根本原因.

图 1 所示为地表覆盖矢量数据的局部示例,图中 C 为一个包含上千个空洞的复杂多边形,图 1(a)中 B、D、E、F 等都是其空洞，其中阴影部分为其他图层图斑,在当前图层中为空白区域. P_1 为增量多边形(图 1(b)),P_1 只与 C 和它的一个空洞多边形 B 和空白区域存在二维交,需要进行更新处理. 但采用现有索引方法,需要计算 P_1 与 C 及其所有空洞多边形的拓扑关系,导致更新处理效率极低. 如果在索引结构中能够存储复杂多边形与其空洞间的包含关系,无关的空洞通过索引过滤掉，那么地表覆盖数据更新效率有望大大提高. 根据上述分析,本文提出一种存储复杂多边形包含关系的空间数据索引方法.

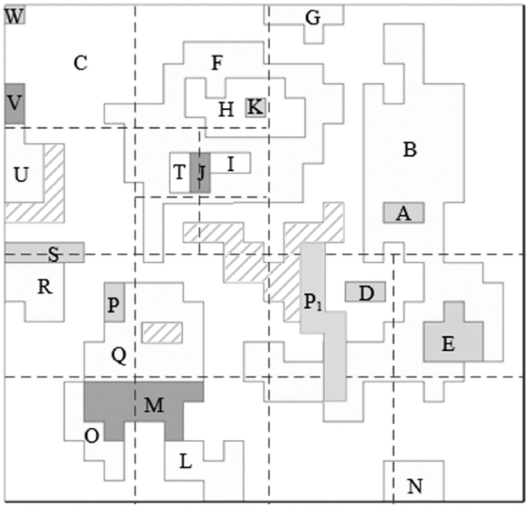
地表覆盖矢量数据嵌套复杂,多边形 MBR 重叠严重,若构建索引 R 树结构,则索引性能不佳[15],同时 R 树索引无法避免地重复存储空间对象，造成空间数据更新时存储的包含关系一致性维护困难. 处理面目标的四叉树结构主要有线性四叉树、PMR 四叉树、CIF 四叉树等结构[9,19-20]. 线性四叉树用自定义大小的网格映射空间目标[21],由于地表覆盖矢量数据面积分布极度不均,难以选择大小合适的网格,同时索引中空间对象也无法避免重复存储;PMR 四叉树索引以线段而非以面目标作为整体概念;CIF 四叉树索引以分层的网格映射空间目标[22],索引结构形态不依赖空间对象插入的顺序,不重复存储空间对象,同时空间数据更新时,结点变更较小[21-23],本文在 CIF 四叉树基础上提出一种存储拓扑包含关系的四叉树空间索引方法.



(a)更新前数据



(b)增量多边形



(c)更新后数据

图 1　地表覆盖复杂多边形增量更新举例
(阴影部分为其他图层图斑)

Fig.1　Example of the incremental updating of land cover complex polygons

1　包含关系四叉树索引的建立

1.1　多边形包含关系的表达

复杂多边形与其空洞多边形之间的嵌套包含关系类似于父子关系，可通过父–子–孙间的序关系来表达多边形之间嵌套包含关系,如图 2 所示.

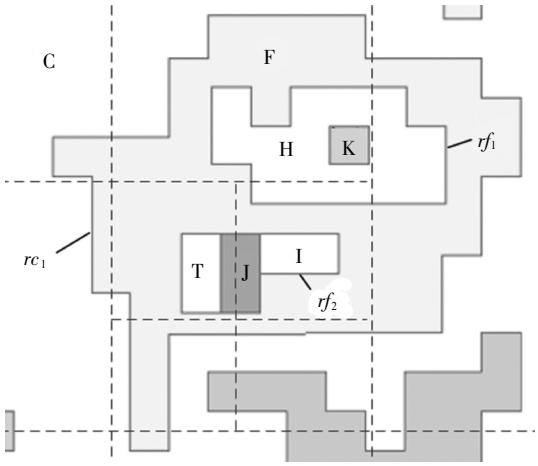


图 2　多边形间的包含关系

Fig.2　Inclusion relations between polygons

图 2 中 H 的父多边形为 F,F 与 H 互为父多边形与子多边形,H 与 C 不存在直接包含关系(H 为 C 的孙子多边形). 子多边形被包围在父多边形的相应内环(Ring in Parent,RIP)中,多边形 F 包含在 C 中的内环 rc_1 中(即 F 的 RIP 为 rc_1),因此,内环是父多边形和子多边形之间的联系，内环嵌套体现多边形之间复杂包含关系. 一个多边形的内环可以被多个子多边形共享,F 的内环 rf_2 包含了 T、J、I 共 3 个子多边形. 内环不是一个独立对象,因此不能直接建立内环对象与其所包围的子多边形对象的对应关系，但通过遍历内环所在多边形的子多边形，判断具有共同 RIP 的子多边形即可确定上述对应关系. 因此，一个多边形的直接包含关系可表达为:{父多边形，在父多边形中相应的环,包含的子多边形}.

根据以上分析,设 CP(Current Polygon)表示当前多边形,PP(Parent Polygon)为 CP 的父多边形,RIPID(Ring in Parent ID) 为 CP 在 PP 中相应的内环序号,CPL(Children Polygon List)为 CP 的子多边形指针数组,为每个内环建立子多边形列表,则四叉树结点中多边形对象的数据结构可表达为:{CP,PP,RIPID,CPL}.

空间数据往往分图层构建,且具有铺盖特征,其中复杂多边形与其空洞多边形可能分别存储在不同

图层中,导致一个图层中存在很多空白区域,如图 1 中的阴影部分 (下同). 当前图层只存储了空白区域的 RIP, 而无相应多边形对象. 由于空白区域的 RIP 不能独立存储为多边形对象, 为完整表达该 RIP 相关的包含关系, 本文引入内环虚拟多边形对象来填满复杂多边形内连续的空白区域, 即内环虚拟多边形对象为一个复杂多边形内空洞边界构建的虚拟对象,其结构为{CP,PP,-RIPID,Ø}. 其中 CP 为内环虚拟多边形,PP 为内环虚拟多边形的父多边形,-RIPID 为该内环在 PP 中的序号(用负号来区别于实际存在的多边形).

为确定多边形间的包含关系, 建立了如下 4 条判别规则. 设多边形 P_1、P_2,若满足以下规则,则 P_1 直接包含 P_2.

规则 1 P_1 有内环;

规则 2 P_2 的 MBR 被 P_1 的 MBR 包含, 同时与 P_1 的某一内环 r 的 MBR 相等 (如图 2 中 F 与 rc_1)或相交(如图 2 中 T 与 rf_2);

规则 3 P_2 上任取一顶点在 r 的环内或环上;

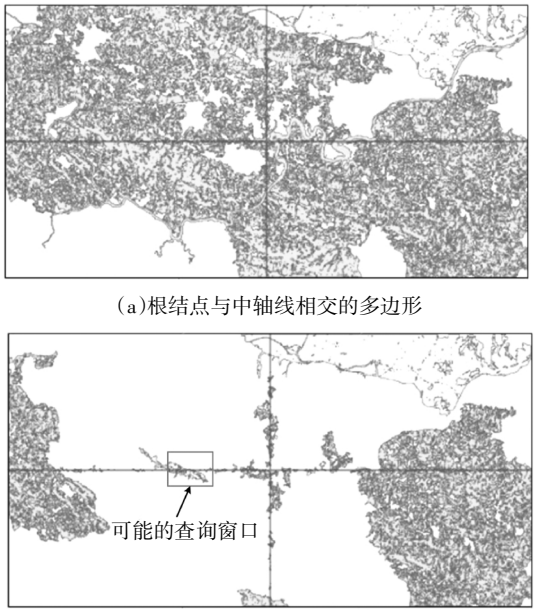
规则 4 不存在 MBR 小于 P_1 的多边形包含 P_2.

上述规则, 每条都是建立在前一条规则的基础上. 其中规则 2 需要遍历 P_1 的内环,对满足规则 2的内环, 再使用规则 3 判别. 若 P_2 的子多边形均为简单多边形, 则前 3 条规则可确定 P_1 与 P_2 的包含关系,否则利用规则 4 进一步约束,以排除嵌套的间接包含关系. 规则 1 判别多边形是否有内环的时间复杂度为常量 $O(1)$,若 P_1 所有内环数的总边数为 e, 则规则 2 遍历 P_1 的内环并判别 MBR 是否相交的主要开销为遍历 P_1 内环计算 MBR,其时间复杂度为 $O(e)$,若 P_2 的 RIP 边数为 a,则规则 3、4 判别的时间开销为判断点是否在多边形内,其复杂度为 $O(a)$[24]. 因此,判别 P_1 包含 P_2 的时间复杂度为 $O(e+a)$.

1.2 对现有四叉树的改进

CIF 四叉树将数据空间递归地划分,直至产生的子象限包含的对象数不大于设定的阈值, 在分解过程中, 所有与象限中轴线相交的对象只与该象限对应的结点相关联, 属于一个结点的矩形不属于任何祖先结点[22]. 索引空间查询过程分为 3 个阶段[21],先经过两次筛选, 然后精确匹配. CIF 四叉树索引查询的第一次筛选判别与查询条件(查询窗口、查询点)相交的四叉树结点,确定候选多边形集,第二次筛选依据候选多边形 MBR 与查询条件是否相交,以缩小结果集的范围, 最后通过精确计算判断二者是否相交,确定查询结果集. Wei 等[10]提出存储结点中所有多边形 MBR 并作为范围 MBR, 以加速筛选第二阶段的候选目标, 但效果并不理想. 第二次筛选过程中,若四叉树结点范围与查询条件相交,仍需要遍历该结点中所有多边形对象. 实际上在四叉树离根近的上层结点中,结点范围大,相应象限的中轴线长,与其相交的多边形数量也很多.

图 3 所示为根结点中存储的多边形示例, 删除与 XY 轴均相交的的复杂多边形后可见与中轴线相交的众多小图斑(图 3(b)). 在索引查询时,当查询窗口仅与 X 负轴相交, 则仍需要遍历根结点中所有多边形,以筛选出目标多边形,尽管大多数多边形与查询窗口相距较远.



(a)根结点与中轴线相交的多边形

(b)图(a)中删除与 XY 轴均相交的多边形

图 3 根结点中存储的多边形示例

Fig.3 Polygons stored in the root of CIF-quadtree inde

为提高第二次筛选的效率, 设四叉树结点对应象限的中轴线交点为坐标原点, 将结点中的多边形按照其与象限中轴线相交的方式不同, 分为只与 X 正轴相交、只与 X 负轴相交、只与 Y 正轴相交、只与 Y 负轴相交以及与 XY 轴都相交 5 种类型(如图 4 所示),并设计了 5 个桶(多边形列表)来存储相应多边形. 筛选时根据桶 MBR 与查询条件的相交情况确定是否遍历桶内多边形. 同时,将与 X 轴相交、XY 轴均相交的桶内多边形根据其 MBR 最小 X 坐标升序排序, 与 Y 轴相交的桶内多边形根据其 MBR 最小 Y 坐标升序排序, 使得索引查询时能在有序序列中筛选可能的查询目标.

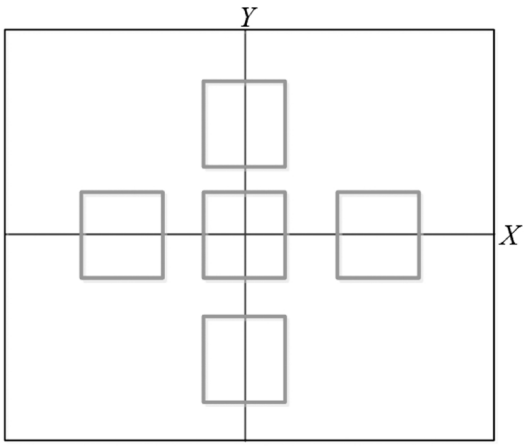


图 4　索引结点中的多边形对象分桶存储

Fig.4　Storage buckets of polygons in the improved index node

根据上述分析，四叉树索引结点的数据结构可表达为：{NID,NMBR,Subtrees,Depth,PPtr,XYL,XpL,XnL,YpL,YnL}. 其中,NID 为结点的 ID,NMBR 为结点的范围 MBR,Subtrees 为子树（四叉）,Depth 为结点层次,PPtr 为父结点指针,XYL、XpL、XnL、YpL、YnL 分别为多边形与中轴线 5 种不同相交方式对应桶. 结点中设计指向父结点的指针是为了方便四叉树中自下而上的遍历.

建立四叉树索引的基本步骤为：

1)根据空间数据范围确定根结点 MBR 并建立根结点，将与根结点中轴线相交的多边形对象按照与中轴线相交的类型有序地插入到相应桶内；

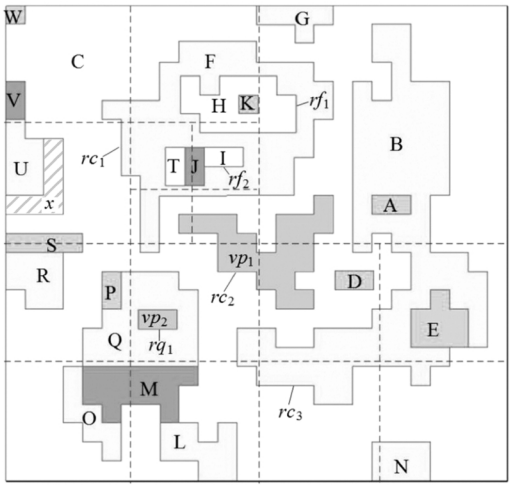
2)根据中轴线将结点等分为 4 个子象限并建立相应子树,将 MBR 完全包含在子象限范围的多边形存储在子树中，并设置子树指向父结点的线索指针 PPtr；

3)对 4 个子树递归地重复步骤 2,直到子树中多边形数量达到结点分裂阈值，获得无包含关系的四叉树索引；

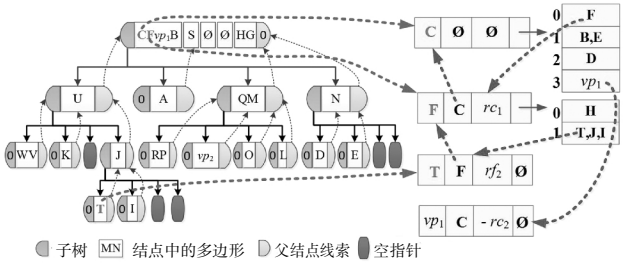
4)对每个多边形对象利用 PPtr 向根搜索并判别父多边形，同时将该多边形加入到父多边形相应内环的 CPL 指针数组中；

5)扫描 CPL,依据内环与子多边形的对应关系，判别并插入虚拟多边形对象.

根据上述建立索引方法对图 1(a)地表覆盖示例数据建立四叉树索引，并以多边形对象 C、F、T 为例表达它们的包含关系,结果如图 5 所示(设分裂阈值 $T_{num}=2$). 图 5(a)中 rc_1、rc_2、rc_3、rf_1、rf_2 及 rq_1 分别为多边形 C、F 及 Q 的内环,其中内环 rc_2 和 rq_1 是其他图层数据,环内空间未铺满,因此建立相应的内环虚拟多边形对象 vp_1 和 vp_2. 虽然 x 区域为其他图层数据，但是没有包含在任何多边形内，无需建立内环虚拟多边形对象. 索引根结点各桶存储情况为:XYL 桶中存储多边形 C、F、vp_1、B;XnL 桶中存储 S;XpL 桶为空;YnL 桶为空;YpL 桶中存储 H、G;图 5(b)中多边形 F 的父多边形指针指向多边形 C,F 存在于 C 中内环对应的子多边形列表中，同时 F 又是多边形 T 的父多边形. 通过 F 可访问父多边形 C 及 F 在 C 中的内环 rc_1，也可访问子多边形 T 及T 在 F 中的内环 rf_2. 图 5(b)中子多边形列表数组与多边形的内环相对应,若多边形无内环(如多边形 T),则该数组为空，否则每个内环对应一个数组元素（子多边形列表). 因此，通过索引可直接获取一个多边形的父多边形、其在父多边形中相应的内环以及该多边形直接包含的子多边形.



(a)地表覆盖数据示例



(b)地表覆盖示例数据的四叉树索引

图 5　改进的四叉树及多边形 C、F、T 之间的包含关系表示

Fig.5　Improved quadtree index and the nested inclusion relations between polygon C,F,T

建立存储包含关系的四叉树索引的时间开销主要包括两部分，即建立无包含关系的索引和确定索引树中多边形对象包含关系. 设多边形数为 n，索引结点数为 N(树深度为 log$N$),则建立四叉树索引的

时间复杂度为 $O(n\cdot\log N)$[19]. 搜索一个多边形的父多边形时,需要在向根的路径上搜索 MBR 与该多边形 MBR 相交的多边形并遍历它的内环. 设四叉树结点中平均多边形数为 n/N,最坏情况下,向根搜索需遍历的多边形数为 $\log N\cdot n$/N, 故建立四叉树索引并确定多边形包含关系的时间总开销为 O ($n\cdot\log N\cdot n/N$)(忽略包括建立无包含关系索引的时间及多边形排序时间等低阶项). 可以看出, 确定多边形包含关系为本文索引建立的主要时间开销, 但索引是一次建立并存储后永久使用, 因此索引建立的代价相对于密集更新业务来说是可接受的.

2 存储包含关系四叉树索引操作

2.1 索引查询

索引查询操作是根据查询条件, 查询与查询点或区域相交的多边形(集),查询操作结果不包括内环虚拟多边形对象. 点查询算法的主要步骤为:

1)自根向下搜索四叉树结点中各桶的 MBR 是否包含该查询点,若包含,则进一步判断桶内多边形 MBR 是否包含查询点,构造候选多边形集;

2)遍历候选集查找第一个满足查询点在外环内的多边形 P;

3)遍历候选集中是否存在 P 的子多边形且满足查询点在该子多边形环外内,则该子多边形为新的 P;

4)重复步骤 3,直至 P 无子多边形,则 P 为查询结果;若 P 为虚拟多边形,则返回空值.

区域查询算法主要步骤为:

1)用与点查询相似方法构造候选多边形集合;

2)遍历候选多边形,将外环与查询窗口相交的多边形(内环虚拟多边形除外)加入查询结果集;

3)遍历候选多边形,若其在候选集中的子多边形(或内环虚拟多边形)的 RIP 与查询窗口相交则将其加入查询结果集;

4)其他候选多边形若窗口任意一点在其外环内且不在其候选集中的子多边形(或内环虚拟多边形) RIP 内,则加入到查询结果集.

查询算法一方面通过桶 MBR 的筛选,能合理避免不必要的索引搜索开销, 另一方面通过存储的多边形间父子包含关系, 避免对候选集中多边形外环及内环逐一判别点在环内的算法开销.

2.2 索引中删除多边形对象

地表覆盖矢量数据的增量更新中, 地块图斑的新增、灭失、分解、合并等各种变更,可以理解为原有地块的消失(删除)和新地块的出现(插入)[25]. 一个增量多边形更新基态数据的过程大致为: 在索引中搜索与增量多边形相交的多边形集合, 逐一更新空间数据,从索引中删除更新前的基态多边形,向索引中插入更新产生的多边形, 并在更新过程中维护包含关系.

从四叉树中删除多边形包括两个任务:1) 删除该多边形的子多边形和父多边形的包含关系;2)删除该多边形对象,若该多边形有父多边形,则建立父多边形的内环虚拟多边形对象并插入到索引中.

从索引中删除多边形(设该多边形为 g)的算法主要步骤为:

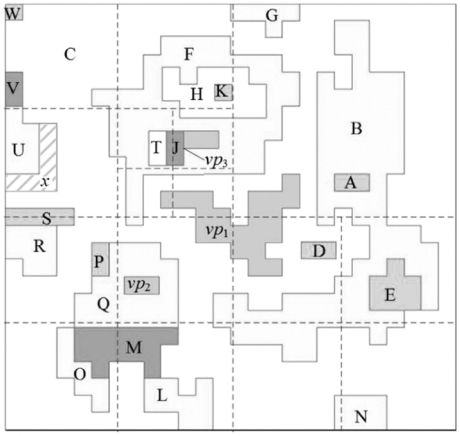
1)从四叉树索引中查询多边形 g 的位置;

2)从 g 的父多边形对象的子多边形列表中删除 g,从 g 的子多边形对象中删除父多边形;

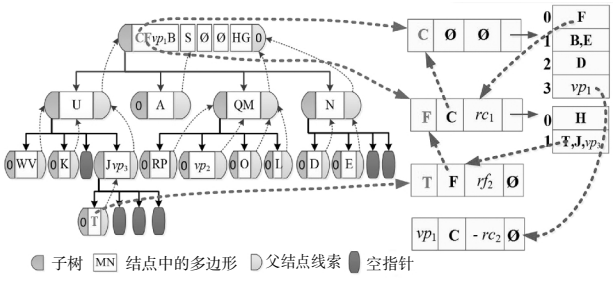
3)从当前结点中删除 g,若父结点不为空,则以 RIP 构建虚拟多边形对象并插入索引;

4)若 g 所在结点为叶结点,则向上递归进行结点合并操作.

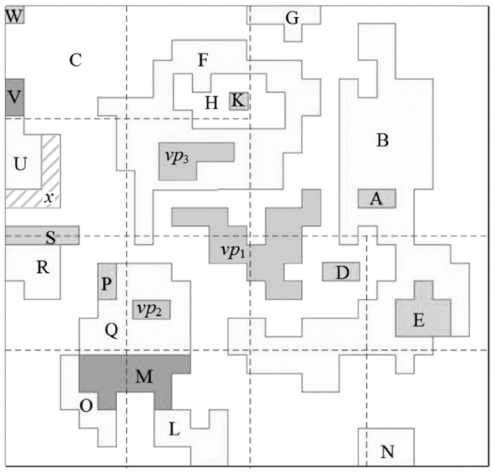
上述算法中的步骤 2) 确保包含关系的一致性,即解除被删除多边形与其他多边形的父子关系. 结点合并操作需要判断 g 的父结点及 g 的兄弟结点中所有多边形数量和是否低于结点分裂阈值 T_{num},若低于 T_{num} 则将这些多边形并入父结点相应桶中, 并将父结点设置为叶结点. 图 1 (a) 中删除多边形 I、T、J 的事件,如图 6(a)(c)所示,设 T_{num}=2,当多边形 I 从索引中删除后,I 有父多边形, 且删除 I 后其 RIP 所围区域未铺满, 需根据其 RIP 建立虚拟多边形对象 rp_3 并插入索引(如图 6(b)所示). 当多边形 T、J 删除后,无需再建立虚拟多边形对象,合并空的叶结点,索引更新后的结果如图 6(d)所示.



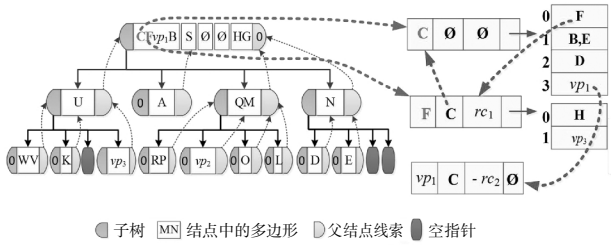
(a)图 1(a)中删除多边形 I



(b)删除多边形 I 后并建立 vp_1 后的索引



(c)继续删除多边形 T、J



(d)继续删除多边形 T、J 后的索引

图 6　图 1(a)中删除多边形 I、T、J,建立内环虚拟多边形 vp_1 后索引的变化

Fig.6　The index structure after merging polygon I and T into polygon J respectively

利用索引进行增量数据更新时，待删除多边形位置已知,步骤 2 删除该多边形的包含关系,即修改其父多边形和子多边形对象的相应属性，步骤 3 依据被删除多边形 RIP 相应子多边形判别是否建立虚拟多边形对象，步骤 4 合并结点是将不多于 T_{num} 个多边形移入父多结点中，以上步骤均可在常数时间内完成,故时间复杂度为 $O(1)$,与现有四叉树索引中删除多边形相比,时间开销增加并不明显.

2.3　索引中插入多边形对象

插入多边形对象前需先在索引中查询插入位置,然后将该对象插入到索引中,并维护相关多边形的包含关系. 插入多边形(设该多边形为 g)的算法主要步骤为:

1)建立多边形 g 的包含关系;

2)在四叉树索引中查询应插入的结点,并将多边形 g 插入到相应的桶中;

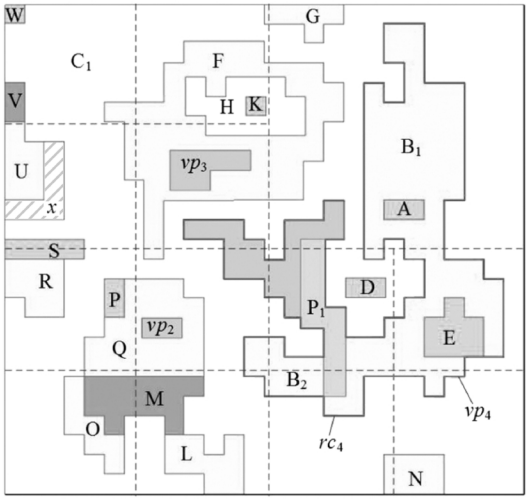
3)若 g 内有更新产生的内环,则判别是否需要从索引中删除相关的虚拟多边形对象;

4)若插入的结点为叶结点,则判断并处理插入操作可能引起的结点分裂.

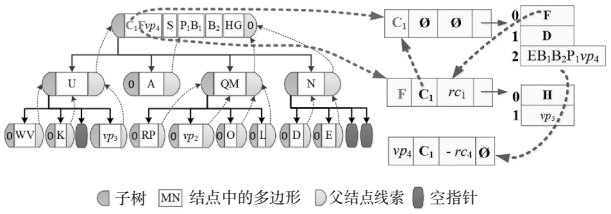
图 7(a)显示图 6(c)中用增量多边形 P_1 更新基态的结果. 增量多边形 P_1 与基态多边形 B、C 及内环虚拟多边形 vp_1 相交，与 B 交于外环，与 C 交于 rc_2 和 B 的 RIP 两个内环. 裁剪 C（只裁剪相交的环)得 C_1 及新内环 rc_4,C_1 继承 C 所有未参与裁剪的内环及子多边形,裁剪 B 得 B_1、B_2,B_1 继承 B 未参与裁剪的内环及子多边形. 通过面积属性知内环 rc_4 未铺满子多边形，建立内环虚拟多边形 vp_4. 最后从索引中删除多边形 C、B、vp_1,将多边形对象 C_1、B_1、B_2、P_1 及 vp_4 插入到索引中,如图 7(b)所示,其中索引根结点各桶存储情况为:XYL 桶存储多边形 C_1、F 以及 vp_4;XnL 桶存储 S;XpL 桶存储 P_1 和 B_1;YnL 桶存储 B_2;YpL 桶存储 H 和 G.

增量更新时需向索引中插入更新后产生的新多边形以及增量多边形. 设与增量多边形相交的多边形集合为 S,S 更新后的集合为 S',S' 与 S 中多边形的父多边形集合的并为 S^+，由于增量多边形的局部性,S、S' 及 S^+中的多边形数量远小于复杂多边形内环数. 若在更新过程中一个多边形的父多边形未被更新时，则更新后的多边形仍被其父多边形直接包含或间接包含,因此,插入到索引中的多边形对象应在 S^+中搜索父多边形并在索引中更新包含关系,其 RIP 为父多边形中与更新操作相关的内环（如 B_1、B_2 的 RIP 为 C_1 的内环 vp_4). 所以增量更新过程中向索引插入多边形时维护包含关系的时间开销来自于构建及搜索集合 S^+(RIP 在增量更新时已确定)，由于 S^+中多边形数量很少,因此在已知 RIP 时,确定插入的多边形的包含关系可在常数时间内完成. 若插入的多边形对象的 RIP 已经被铺满，则应删除该内环的虚拟多边形对象. 对 RIP 内的多边形面积属性求和并判断其是否与 RIP 所围面积相等，确定是否需要删除虚拟多边形,因此其时间复杂度为 $O(b)$,其中 b 为 RIP 的边数. 由插入多边形对象引起的结点分裂的时间复杂度为 $O(T_{num})$,其中 T_{num} 为分裂阈值常量. 在索引中查询多边形插入位置的时间开销为

遍历从根向叶子的一条路径并按顺序插入多边形，这与现有四叉树插入时间一致,时间复杂度为 $O(n/N\cdot\log N)$，其中 n/N 为结点平均多边形数. 故索引中插入多边形的时间复杂度为 $O\ (n/N\cdot\log N+b)$. 与现有四叉树索引相比，本文索引插入多边形时间开销增加了包含关系维护的时间.



(a)增量多边形 P1 更新图 6(c)



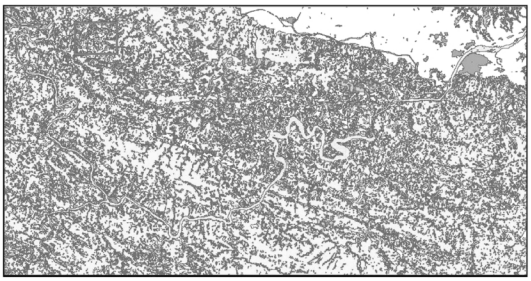
(b)增量更新后索引的变化

图 7 增量多边形 P_1 更新图 6(c)数据后索引的变化

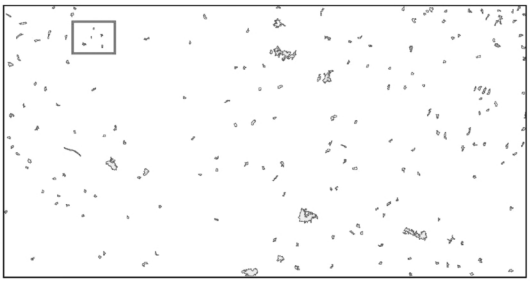
Fig.7 The index structure after updating the data in Fig.6(c) with incremental polygon P_1

3 实验与比较

为验证本文存储包含关系四叉树索引的有效性，以地表覆盖矢量数据增量更新为例进行了实验验证. 实验数据来源于 30 m 全球地表覆盖数据(GlobeLand30),选取陕西省 2000 年和 2009 年两景轨道号为 127034 的 Landsat ETM+/TM 30 m 分辨率遥感影像数据. 影像覆盖范围为北纬 32.432°~32.760°,东经 108.384°~109.100°，面积为 2 373.1 km². 用比值法、NDVI 差值法、PCA 差异法求得的结果影像作为初始变化信息并对其分类，为提高分类精度，采用 2009 年样本数据对 2009 年影像进行分类，然后以 2009 年数据为基态数据与 2000 年影像进行变化信息提取,获得增量数据. 采用项目组开发的分类后遥感影像自动矢量化与伪变化剔除组件自动矢量化并剔除伪变化，生成原始基态矢量数据和增量矢量数据如图 8 所示. 图 8(b)中方框内为图 1 所在区域. 该矢量数据采用地表覆盖数据常用的 Shapefile 格式存储[7],多边形总数为 104 230 个,数据中存在包含大量空洞的复杂多边形,其中超过 1 000 个空洞的复杂多边形 6 个，最复杂的多边形包含 5 573 个空洞. 增量矢量数据为简单即不包含空洞的多边形. 为实验需要，对获取的矢量基态数据分别以不同最小面积剔除合并,得到不同多边形数量的矢量基态数据. 实验硬件环境为联想杨天 A8000 微型计算机,CPU 为 i7-7700,内存 16 GB.



(a)2009 年地表覆盖基态矢量数据



(b)2000 年地表覆盖增量矢量数据

图 8 原始实验矢量数据

Fig.8 Experimental vector land cover data before updating

本文以 VS2013 为平台开发了增量更新系统. 根据提出的索引方法，对不同多边形数量的基态数据建立索引,索引建立所用时间与文献[10]CIF 四叉树索引对比如表 1 所示.

表 1 索引建立的时间与空间消耗

Tab.1 Costs for index creation

基态多边形数（最大环数）	数据大小/kB	建立时间/s		索引大小/kB	
		本文索引	CIF 索引	本文索引	CIF 索引
6 585(614)	8 463	0.667 1	0.112 9	224	144
12 166(1 079)	11 024	1.640 6	0.241 9	519	259
26 003(2 140)	15 056	5.070 4	0.766 5	915	513
56 087(3 817)	20 540	16.900 4	2.968 2	2 319	1 404
104 230(5 573)	27 962	43.169 1	9.489 1	4 027	2 529

表 1 中基态多边形“最大环数”是指基态多边形中最复杂的多边形所具有的空洞数，建立本文索引时四叉树结点分裂阈值为 30. 阈值是由四叉树索引插入和删除操作的频率根据经验来设定，阈值过大会使四叉树查询接近线性查询，过小会造成插入和删除时结点频繁分裂或合并，增加维护结点的时间开销. 本文索引建立的主要时间和空间开销为包含关系的建立与存储，因此索引建立时间相较于现有四叉树索引建立,增加了建立包含关系的时间开销,随着数据量的增加索引建立的时间也会延长，但是索引建立时间总体不长（本文实验超过 10 万个多边形,一般个人计算机建立索引的时间不超过 1min). 由于存储包含关系及内环虚拟多边形对象空间开销较大，平均约占原始数据的 8%,约为 CIF 索引的 1.5~2 倍. 由于索引结点中的对象存储了包含关系，使索引结构变得复杂,从本质上看,是增加了索引对象空间数据表的列数,可以通过截断数据表,增加连接字段来减小索引结构复杂带来的影响.

为验证本文索引的查询性能,建立文献[10] CIF 四叉树索引并开展查询和更新对比实验. 查询实验分为点查询和区域查询，先在实验数据区域内随机生成 50 个查询点和 50 个大小不等位置随机的查询窗口,然后在原始基态矢量数据(多边形数量为 104 230 个)中运用上述两种索引分别做点查询和区域查询，并记录查询的时间和. 查询实验共进行 5 次,实验结果如表 2 所示,表中时间为平均每个点查询或区域查询所需时间(ms).

表 2　利用本文索引与 CIF 四叉树索引的查询时间对比

Tab.2　Query time with the impoved index comparing with CIF quadtree index　　ms

查询实验轮次	点查询时间		区域查询时间	
	本文索引	CIF 索引	本文索引	CIF 索引
第一次	3.261	7.300	56.309	102.695
第二次	2.942	6.552	55.674	81.362
第三次	3.441	6.413	51.864	79.996
第四次	3.032	7.859	64.634	92.702
第五次	2.882	7.021	51.239	84.249
平均	3.112	7.029	55.944	88.201

由表 2 可知,本文索引比文献[10] CIF 索引的查询效率有显著提高,这得益于索引结点分桶存储多边形对象,加速了查询过程第二次筛选.

为验证索引操作的有效性,利用图 8(b)增量数据对不同矢量基态数据增量更新,其中增量数据共 181 个多边形. 更新过程中,记录索引操作的次数,并与文献[10] CIF 四叉树索引方法进行对比. 实验结果如表 3 所示,其中“更新复杂多边形次数”是指增量数据更新基态复杂多边形的次数;复杂多边形是指内环空洞数超过 300 的多边形;“最大环数”是指基态多边形中最复杂的多边形所具有的空洞数;“更新时间”是指分裂阈值为 30 时利用相应空间索引增量更新的时间(s),包括删除增量更新前基态多边形的时间、插入新多边形(被删除的基态多边形与增量多边形的差多边形)的时间以及完成索引维护的时间.

表 3　利用本文索引与利用 CIF 四叉树索引增量更新时间对比

Tab.3　Cost time comparing between our improved index method and CIF quadtree index in incremental updating

基态多边形数（最大环数）	更新复杂多边形次数	删除多边形次数	插入多边形次数	CIF 索引更新时间 $T1$/s	本文索引更新时间 $T2$/s	$T1/T2$
6 585(614)	84	824	381	77.107 4	26.622 5	2.9
12 166(1 079)	118	951	455	99.348 5	29.618 5	3.4
26 003(2 140)	122	1 120	595	144.397 2	34.626 4	4.2
56 087(3 817)	124	1 378	869	208.726 7	39.866 1	5.2
104 230(5 573)	128	1 609	1 228	286.134 6	46.021 4	6.2

由表 3 可知，本文提出的索引开展增量更新实验能显著提高更新的效率，所需时间平均约为文献[10]索引增量更新时间的 25%. 本文索引更新时数据量越大优势越明显，根本原因在于本文索引存储的包含关系使得增量更新时能快速确定基态多边形与增量多边形相交的内环,不受多边形复杂程度(空洞多少)的影响,有效避免不必要的时间开销,提高了更新效率；另外分桶有序存储对提高查询效率累积

效果明显.

4 结论与讨论

多边形之间的多层嵌套包含关系反映了地表覆盖矢量数据的复杂程度，考虑多边形之间的包含关系，能显著提高包含大量空洞的复杂多边形增量更新效率. 目前空间数据索引一般只用来提高空间查询效率，不能反映包含空洞的复杂多边形及其空洞多边形之间的嵌套关系，在地表覆盖变化冲突检测与更新处理中对于包含成千上万个空洞的复杂多边形来说，增量更新计算效率不能满足实际应用需求. 本文针对这一问题在已有四叉树基础上提出一种存储拓扑包含关系的四叉树空间索引方法，为提高结点搜索效率，该方法根据结点中存储的多边形对象与四叉树结点相应的象限中轴线相交的不同方式分为仅与 X 正轴、X 负轴、Y 正轴、Y 负轴相交以及与 XY 轴都相交 5 种类型,根据这 5 种类型将多边形分别排序并存储在索引结点的相应桶中，通过桶筛选减少索引查询过程中空间对象匹配次数，同时也避免了多边形对象在索引中重复存储；本文在多边形对象中存储多边形之间的父子包含关系，建立存储复杂多边形包含关系的空间四叉树索引，并设计了存储包含关系索引的查询、插入、删除等算法;引入内环虚拟多边形，解决了复杂多边形与其空洞多边形分层存储导致的复杂多边形包含关系不完整问题. 最后在 VS2013 平台上实现该索引方法,运用提出的索引操作算法开展了基于该索引方法的地表覆盖矢量数据增量更新实验，实现了地表覆盖矢量数据批量增量更新，并维护了索引更新中包含关系的一致性，验证了所提索引方法的有效性. 实验结果表明，利用本文索引对复杂地表覆盖数据进行空间查询的效率比现有四叉树索引效率显著提高，在本文实验中增量更新的效率为文献 [10] 四叉树索引方法的 2.9~6.2 倍,且随着数据量的增加效率提高更明显.

尽管本文存储拓扑包含关系的四叉树空间索引方法是针对地表覆盖数据更新提出的，该方法同样可用于土地利用等包含大量复杂多边形的应用领域；由于本文索引中存储了复杂多边形与其空洞多边形之间的包含关系，有利于包含关系的查询与统计分析,如湘江有多少个岛等. 由于本文索引需要建立多边形间的包含关系，目前该索引建立时间要比现有四叉树索引建立时间长，索引更新的效率仍有提升空间；本文实验假设在数据更新等应用中索引结构存储于内存中，尚需探索特大数据情况下将索引结构存储在外存数据库的情形. 因此后续研究工作主要包括提高索引更新的效率及在特大数据情况下本文索引方法的适应性探索.

参考文献

[1] CHEN J,CHEN J,LIAO A P,*et al*. Global land cover mapping at 30 m resolution:a POK-based operational approach [J]. ISPRS Journal of Photogrammetry and Remote Sensing,2015,103:7—27.

[2] CHEN J,CAO X,PENG S,*et al*. Analysis and applications of Global Land 30:a review [J]. ISPRS International Journal of Geo-Information,2017,6(8):230.

[3] GONG P,LIU H ,ZHANG M N,*et al*. Stable classification with limited sample:transferring a 30-m resolution sample set collected in 2015 to mapping 10-m resolution global land cover in 2017 [J]. Science Bulletin,2019,64(6):370—373.

[4] 陈斐,陈军,武昊,等. 基于景观形状指数的地表覆盖检验样本自适应抽样方法[J]. 中国科学:地球科学,2016,46(11):1413—1425.
CHEN F,CHEN J,WU H,*et al*. A landscape shape index-based sampling approach for land cover accuracy assessment [J]. Science China Earth Sciences,2016,46(11):1413—1425. (In Chinese)

[5] 陈军,武昊,李松年. 全球地表覆盖领域服务计算的研究进展--以 GlobeLand 30 为例[J]. 测绘学报,2017,46(10):1526—1533.
CHEN J,WU H,LI S N. Research progress of global land domain service computing:take GlobeLand 30 as an example [J]. Acta Geodaetica et Cartographica Sinica,2017,46(10):1526—1533.(In Chinese)

[6] 魏东升，周晓光. 顾及纹理特征贡献度的变化影像对象提取算法[J]. 测绘学报,2017,46(5):605—613.
WEI D S,ZHOU X G. Changed image objects extraction algorithms considering texture feature contribution [J].Acta Geodaetica et Cartographica Sinica,2017,46(5):605—613. (In Chinese)

[7] 周晓光,汪红松,吴志强. 引入二维交细分类型的地表覆盖矢量数据增量更新[J]. 测绘学报,2017,46(1):114—122.
ZHOU X G,WANG H S,WU Z Q. An incremental updating method for land cover database using refined 2-dimensional intersection type [J].Acta Geodaetica et Cartographica Sinica,2017,46(1):114—122. (In Chinese)

[8] XING H F,MENG Y,WANG Z X,*et al*. Exploring geo-tagged photos for land cover validation with deep learning [J]. ISPRS Journal of Photogrammetry and Remote Sensing,2018,141:237—251.

[9] HJALTASON G R,SAMET H . Speeding up construction of PMR quadtree-based spatial indexes [J]. The VLDB Journal,2002,11(2):109—137.

[10] WEI Y,TANAKA S . Performance improvement of MX-CIF quadtree by reducing the query results [J]. International Journal of Computer Theory & Engineering,2012,4(6):902—906.

[11] GUTTMAN A . R-trees:a dynamic index structure for spatial searching [C]// ACM SIGMOD International Conference on Management of Data. Boston:MCM,1984:47—57.

[12] JIN P Q,XIE X K,WANG N,*et al*. Optimizing R-tree for flash memory [J]. Expert Systems with Applications,2015,42(10):4676—4686.

[13] SELLIS T K,ROUSSOPOULOS N,FALOUTSOS C. The R+-tree:a dynamic index for multi-dimensional objects [C]// International Conference on Very Large Data Bases. Brighton:Margan kaufmann,1987:507—518.

[14] BECKMANN N,KRIEGEL H P,SCHNEIDER R,*et al*. The R*-tree:an efficient and robust access method for points and rectangles [C]// ACM SIGMOD international conference on management of data. Atlantic City,New Jersey:ACM,1990:322—331.

[15] ROUMELIS G,VASSILAKOPOULOS M,CORRAL A,*et al*. Efficient query processing on large spatial databases:A performance study [J]. Journal of Systems and Software,2017,132:165—185.

[16] JI C Q,LI Z Y,QU W Y,*et al*. Scalable nearest neighbor query processing based on inverted grid index [J]. Journal of Network and Computer Applications,2014,44:172—182.

[17] CHEN H L,CHANG Y I. All-nearest-neighbors finding based on the hilbert curve [J]. Expert Systems with Applications,2011,38(6):7462—7475.

[18] 向隆刚,高萌,王德浩,等. Geohash-Trees:一种用于组织大规模轨迹的自适应索引[J]. 武汉大学学报(信息科学版),2019,44(3):436—442.
XIANG L G,GAO M,WANG D H,*et al*. Geohash-Trees:an adaptive index which can organize large-scale trajectories [J]. Geomatics and Information Science of Wuhan University,2019,44(3):436—442.(In Chinese)

[19] SAMET H . The design and analysis of spatial data structures Addison-Wesley Series in Computer Science [M]. Massachusetts:Addison-Wesley,1990:199—209.

[20] SAMET H. Foundations of multidimensional and metric data structures[M]. San Mateo:Morgan Kaufmann,2006:466—479.

[21] KOTHURI R K V,RAVADA S,ABUGOV D. Quadtree and R-tree indexes in oracle spatial:a comparison using GIS data [C]// Proceedings of the 2002 ACM SIGMOD international conference on Management of data. Madison Wisconsin:ACM,2002:546—577.

[22] 郭薇,郭菁,胡志勇. 空间数据库索引技术[M]. 上海:上海交通大学出版社,2006:100—103.
GUO W,GUO J,HU Z Y. The technology of spatial database index [M]. Shanghai:Shanghai Jiao Tong University Press,2006:100—103.(In Chinese)

[23] ZIMMERMANN R,KU W S,CHU W C. Efficient query routing in distributed spatial databases [C]// ACM International Workshop on Geographic Information Systems.Washington D C:ACM,2004:176—183.

[24] HU Y,RAVADA S,ANDERSON R. Geodetic point-in-polygon query processing in oracle spatial [C]//International Symposium on Spatial and Temporal Databases. Minneapolis:Springer,2011:297—312.

[25] 张新长,郭泰圣,唐铁. 一种自适应的矢量数据增量更新方法研究[J]. 测绘学报,2012,41(4):613—619.
ZHANG X C,GUO T S,TANG T. An adaptive method for incremental updating of vector data [J].Acta Geodaetica et Cartographica Sinica,2012,41(4):613—619. (In Chinese)