# 三维空间中面对象相邻关系的判断算法

孙晓明[1,2],吉根林[1,2]

(1. 南京师范大学 计算机科学与技术学院,江苏 南京 210097;
2. 江苏省信息安全保密技术工程研究中心,江苏 南京 210097)

[摘要] 提出了三维空间中面对象相邻关系的判断算法 CRAR 与算法 CRAR-DF. 计算空间面对象与其他空间对象之间的距离,如果该距离小于给定的阈值,则认为面对象与该对象相邻. 算法 CRAR-DF 在算法 CRAR 的基础上利用距离函数进行相邻关系的快速判断. 实验结果表明,算法 CRAR 和算法 CRAR-DF 能够准确地判断面对象的相邻关系,且 CRAR-DF 具有较高的效率.

[关键词] 三维拓扑分析,相邻关系,面对象
[中图分类号] TP391 [文献标志码] A [文章编号] 1672-1292(2011)02-0073-06

# Algorithms for Computing Region Adjacent Relations in 3D Space

Sun Xiaoming, Ji Genlin

(1. School of Computer Science and Technology, Nanjing Normal University, Nanjing 210097, China;
2. Jiangsu Research Center of Information Security and Privacy Technology, Nanjing 210097, China)

**Abstract:** Algorithms CRAR and CRAR-DF are proposed for computing region adjacent relations in 3D space in this paper. Region objects and other spatial objects are adjacent if their distance is less than threshold value. Based on CRAR, CRAR-DF judges adjacent relations rapidly according to distance function. The experimental results show that algorithms CRAR and CRAR-DF are effective and CRAR-DF is more efficient.

**Key words:** 3D topological analysis, adjacent relation, region

空间关系是地理信息系统(Geographical Information System,GIS)的重要理论问题之一. 空间拓扑关系是空间关系的主要研究内容. 空间拓扑关系指的是拓扑变换(平移、旋转、缩放等)下的拓扑不变量[1]. 拓扑关系在空间数据建模、空间查询、分析、推理、制图综合、图像检索和相似性分析等过程中起着重要的作用[2]. 相邻关系是一种基本的拓扑关系. GIS 最初使用二维投影方式解决对现实世界各类三维空间对象的表达,这种方式虽然简化了空间信息理解与表达的过程,却损失了空间信息量,是以牺牲空间信息的真实性和完整性为代价的[3]. 因此,研究三维拓扑关系有着重要的意义.

空间对象相邻关系的判断主要是计算空间对象之间的距离. 两个点对象之间的距离可以直接采用曼哈顿距离、欧氏距离等标准距离. 而线对象(公路、河流等)和面对象(学校、行政区等)简单地以中心点或其他点来代表是不恰当的. 文献[4]提出了 3 种不同的方法来计算两个多边形的距离. 第一种方法基于计算两个多边形间的精确距离进行判断,也就是计算这两个多边形的点与点距离的最小值. 第二种方法使用多边形顶点间的最短距离来近似多边形的距离. 第三种方法使用多边形的最小外包矩形的距离来近似. 文献[5]提出了二维空间中线对象相邻关系的判断算法. 文献[6]通过先考虑二维空间中两任意实体之间不同的相对位置关系,再利用构建 Delaunay 三角网寻找两者的邻近区域,从而计算出两者之间的最近距离. 三维空间中异面情况的存在使得空间对象间距离关系的描述和计算更为复杂,目前对三维空间中相邻关系的研究较少.

本文提出了三维空间中面对象相邻关系的判断算法 CRAR. 此外,在 CRAR 的基础上利用距离函数进

收稿日期: 2011-03-16.
基金项目: 国家自然科学基金(40871176).
通讯联系人: 吉根林,博士,教授,博士生导师,研究方向: 数据挖掘技术及其应用. E-mail: glji@njnu.edu.cn

行相邻关系的快速判断,提出了算法 CRAR-DF. 实验结果表明,上述两个算法能够准确地判断面对象的相邻关系,CRAR-DF 能有效提高算法的执行效率.

## 1　相关定义

**定义1**　点[7]:一个三维空间中的点对象 $P$ 由单个点$(x, y, z)$ 表示,可表示树或商店的位置.

**定义2**　线[7]:一个三维空间中的线对象 $L$ 由一个顶点序列$\{P_1, P_2, \cdots, P_m\}$ 表示,可以用来表示河流、道路等.

**定义3**　面[7]:一个三维空间中的面对象 $S$ 是由一组点对象$\{P_1, P_2, \cdots, P_n\}$ 围成的闭合区域,可表示湖泊、住宅区等.

**定义4**　相邻关系:空间对象 $O_1$、$O_2$ 的距离在$(0, D_{\max}]$ 之间,称 $O_1$、$O_2$ 相邻,其中 $D_{\max}$ 为满足相邻关系的空间对象间距离的最大值. 这里的距离指的是欧式距离,且为两个空间对象上的点间距离的最小值.

**定义5**　包围盒[8](Axis-Aligned Bounding Boxes):三维空间中一个给定对象 $O$ 的包围盒 $AABB$ 被定义为包含该对象且各边平行于坐标轴的最小的六面体,表示为它对角的两个顶点 $l$ 和 $h$ 的组合,$AABB_O = (l_x, l_y, l_z, h_x, h_y, h_z)$,其中 $l_x \leqslant h_x, l_y \leqslant h_y, l_z \leqslant h_z$.

**定义6**　最小距离[8](Minimum Distance,MINDIST):两个 $AABB$ 的最小距离就是这两个六面体边界上所有点之间距离中的最小者,也就是它们所包含的空间对象间距离的下界. 给定 $AABB_R$ 和 $AABB_S$,它们之间的最小距离定义为

$$\mathrm{MINDIST}(AABB_R, AABB_S) = \sqrt{p_x^2 + p_y^2 + p_z^2},$$

其中,$p_x = \begin{cases} AABB_R.h_x - AABB_S.l_x & \text{if } AABB_R.h_x < AABB_S.l_x; \\ AABB_R.l_x - AABB_S.h_x & \text{if } AABB_R.l_x > AABB_S.h_x; \\ 0, & \text{otherwise.} \end{cases}$

$p_y$、$p_z$ 的定义与 $p_x$ 类似.

## 2　三维空间面对象相邻关系判断算法 *CRAR*

### 2.1　点-线距离计算方法

计算点对象到线对象的距离时就是计算点对象到组成线对象的每条线段的距离的最小值.

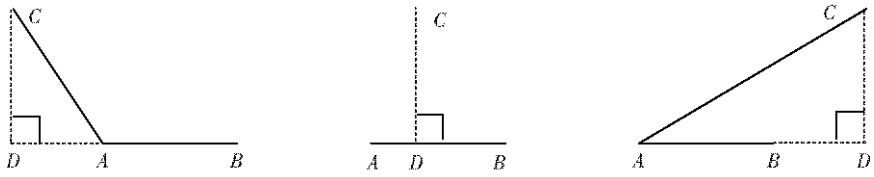


图1　点与线段的距离计算

Fig.1　Distance computation of point and segment

点对象与线段的距离可以用点乘法求得. 如图1所示,求点 $C$ 到线段 $AB$ 的最小距离. 根据 $\lambda = \dfrac{\boldsymbol{AB} \cdot \boldsymbol{AC}}{|\boldsymbol{AB}|^2}$ 得到向量 $\boldsymbol{AC}$ 在向量 $\boldsymbol{AB}$ 上的投影 $\boldsymbol{AD}$ 与 $\boldsymbol{AB}$ 的比值,若 $\lambda < 0$,说明点 $D$ 在线段 $BA$ 延长线上(图1(a)),则距离为 $|\boldsymbol{AC}|$;若 $\lambda > 1$,说明点 $D$ 在线段 $AB$ 延长线上(图1(c)),则距离为 $|\boldsymbol{BC}|$;否则,点 $D$ 在线段 $AB$ 上(图1(b)),距离为 $|\boldsymbol{CD}|$,由 $\boldsymbol{AD} = \lambda\boldsymbol{AB}$ 及勾股定理得 $|\boldsymbol{CD}| = \sqrt{|\boldsymbol{AC}|^2 - (\lambda|\boldsymbol{AB}|)^2}$. 算法描述如下:

**算法1**　计算点对象与线段之间距离的算法 PSegDis$(C, AB)$.

**输入:** 点对象 $C(x, y, z)$,线段 $AB$ 的端点坐标 $A(x, y, z)$,$B(x, y, z)$.

**输出:** 点对象与线段的距离.

**步骤:**

(1)　$ab.x = B.x - A.x$; $ab.y = B.y - A.y$; $ab.z = B.z - A.z$;　// 求向量 $\boldsymbol{AB}$

(2) $ac.x = C.x - A.x$; $ac.y = C.y - A.y$; $ac.z = C.z - A.z$; // 求向量 $\boldsymbol{AC}$

(3) namta = ($ab.x * ac.x + ab.y * ac.y + ab.z * ac.z$) /($ab.x * ab.x + ab.y * ab.y + ab.z * ab.z$); // 求 λ

(4) if( namta < 0) //$D$ 在 $BA$ 延长线上

(5) dis = sqrt($ac.x * ac.x + ac.y * ac.y + ac.z * ac.z$);

(6) else if( namta > 1){ //$D$ 在 $AB$ 延长线上

(7) $bc.x = C.x - B.x$; $bc.y = C.y - B.y$; $bc.z = C.z - B.z$; // 求向量 $\boldsymbol{BC}$

(8) dis = sqrt($bc.x * bc.x + bc.y * bc.y + bc.z * bc.z$);

}

(9) else{ //$D$ 在 $AB$ 间

(10) dis = sqrt($ac.x * ac.x + ac.y * ac.y + ac.z * ac.z -$

(11) namta * namta * ($ab.x * ab.x + ab.y * ab.y + ab.z * ab.z$));

}

(12) return dis; // 返回点到线段的距离

## 2.2 线 - 线距离计算方法

空间线对象与线对象的距离可以通过计算线对象与构成另一个线对象的每条线段的距离,取最小值. 两个线段的距离计算方法如下: 如图 2 所示,线段 1 的端点为 $P_1$、$P_2$,所在直线为 $L_1$,线段 2 的端点为 $Q_1$、$Q_2$,所在直线为 $L_2$,如果 $L_1$、$L_2$ 共面,分别求两线段的端点到另一个线段的距离,取最小值. 如果异面,求出它们和公垂线的交点 $S_1$、$S_2$,$S_1$、$S_2$ 间的距离记为 $h_1$,如果 $S_1$ 在线段 1 上且 $S_2$ 在线段 2 上,则线段 1 和线段 2 的距离为 $h_1$,否则,过 $S_2$ 作平行于 $L_1$ 的直线 $L_1'$,显然 $L_1'$、$L_2$ 共面且垂直于公垂线,计算出线段 1 在 $L_1'$ 上的对应线段 3,端点为 $P_1'$、$P_2'$,计算线段 3 和线段 2 的距离,记为 $h_2$,则两线段的距离为 $\sqrt{h_1^2 + h_2^2}$. 算法描述如下:

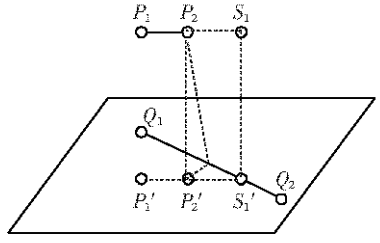


图 2 线段与线段的距离计算

Fig.2 Distance computation of two segments

**算法 2** 计算线段与线段之间距离的算法 SegSegDis($P_1P_2$,$Q_1Q_2$).

**输入:** 线段 $P_1P_2$ 的端点坐标 $P_1(x,y,z)$,$P_2(x,y,z)$,线段 $Q_1Q_2$ 的端点坐标 $Q_1(x,y,z)$,$Q_2(x,y,z)$.

**输出:** 线段 $P_1P_2$ 与线段 $Q_1Q_2$ 之间的距离.

**步骤:**

(1) if( SegSegIntersect($P_1P_2$,$Q_1Q_2$)) // 线段 $P_1P_2$、$Q_1Q_2$ 相交,距离为 0

(2) return 0;

(3) else{

(4) if( IsCoplanar($P_1P_2$,$Q_1Q_2$)){ //线段 $P_1P_2$、$Q_1Q_2$ 共面,距离为 4 个端点到另一个线段距离的最小值

(5) dis = min( PSegDis($P_1$,$Q_1Q_2$), PSegDis($P_2$,$Q_1Q_2$), PSegDis($Q_1$,$P_1P_2$), PSegDis($Q_2$,$P_1P_2$));

}

(6) else{ //$P_1P_2$、$Q_1Q_2$ 异面

(7) cp = ComputeCommonPerpendicular($P_1P_2$,$Q_1Q_2$); // 计算 $P_1P_2$,$Q_1Q_2$ 的公垂线

(8) $S_1$ = ComputePointOfIntersection( cp,$P_1P_2$); // 计算公垂线与 $P_1P_2$ 的交点

(9) $S_2$ = ComputePointOfIntersection( cp,$Q_1Q_2$); // 计算公垂线与 $Q_1Q_2$ 的交点

(10) dis1 = PPDis($S_1$,$S_2$); // 计算公垂线段的长度

(11) if( PointOnSeg($S_1$,$P_1P_2$) &&PointOnSeg($S_2$,$Q_1Q_2$))

(12) dis = dis1; //$S_1$ 在线段 $P_1P_2$ 上且 $S_2$ 在线段 $Q_1Q_2$ 上,距离为公垂线段的长度

(13) else{

(14) $P_1'$ = ComputeCorrespondentPoint($P_1$,$S_1$,$S_2$); // 计算 $P_1$ 的对应点 $P_1'$

(15) $P_2'$ = ComputeCorrespondentPoint($P_2$,$S_1$,$S_2$); // 计算 $P_2$ 的对应点 $P_2'$
(16)　dis2 = min(PSegDis($P_1'$,$Q_1Q_2$), PSegDis($P_2'$,$Q_1Q_2$),
(17)　PSegDis($Q_1$,$P_1'P_2'$), PSegDis($Q_2$,$P_1'P_2'$)) // 线段 $P_1'P_2'$、$Q_1Q_2$ 共面,距离为4个端点到另一个线段距离的最小值
(18)　dis = sqrt(dis1 · dis1 + dis2 · dis2);
}
}
(19) return dis; // 返回线段到线段的距离
}

### 2.3 点 - 面距离计算方法

空间点对象与面对象的距离计算如下:首先计算点对象在面对象所在平面上的投影,判断投影是否在该面对象上,如果在(图3中 $Q_1$),则点面距离为该点与其投影间的距离,否则,计算该点(图3中 $P_1$) 与构成面的每条线段的距离,取最小值. 算法描述如下:

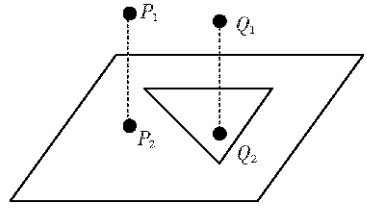


图3　点对象与面对象的距离计算

Fig.3　Distance computation of point and region

**算法3　计算点对象与面对象之间距离的算法** PSDis($P$,$S$).

**输入:** 点对象 $P$,面对象 $S$.

**输出:** 点对象 $P$ 与面对象 $S$ 之间的距离.

**步骤:**

(1) if(PSContain($P$,$S$)) // 面包含点,距离为0
(2) return 0;
(3) else{
(4)　$Q$ = ComputeProjection($P$,$S$); // 计算点在面对象所在平面的投影
(5)　if(PSContain($Q$,$S$)) // 面包含投影,距离为点对象与投影的距离
(6)　dis = PPDis($P$,$Q$);
(7)　else{
(8)　dis = Infinity;
(9)　for each segment Seg of $S$ // 计算点到面的每条线段的距离,取最小值
(10)　dis1 = PSegDis($P$, Seg);
(11)　if(dis1 < dis) dis = dis1;
}
(12) return dis;
}

### 2.4 线 - 面距离计算方法

空间线对象与面对象的距离就是计算面对象与构成线对象的每条线段的距离,取最小值. 线段与面对象的距离计算如下:计算线段的端点在面对象所在平面上的投影,判断投影是否在该面对象上,如果在,计算点与投影间的距离,然后计算线段与构成面对象的每条线段的距离,取上述距离的最小值. 算法描述如下:

**算法4　计算线段与面对象之间距离的算法** SegSDis($P_1P_2$,$S$).

**输入:** 线段 $P_1P_2$ 的端点坐标 $P_1(x,y,z)$,$P_2(x,y,z)$,面对象 $S$.

**输出:** 线段 $P_1P_2$ 与面对象 $S$ 之间的距离.

**步骤:**

(1) if(LSContain($P_1P_2$,$S$) &&LSIntersect($P_1P_2$,$S$)) // 包含或者相交
(2)　return 0; // 距离为0
(3) dis = Infinity;

```
(4) Q1 = ComputeProjection(P1,S);   // 计算 P1 在面对象所在平面的投影
(5)     if(PSContain(Q1,S)){   // 面包含投影,计算点对象与投影的距离
(6)     dis1 = PPDis(P1,Q1);
(7)     if(dis1 < dis) dis = dis1;
    }
(8) Q2 = ComputeProjection(P2,S);   // 计算点在面对象所在平面的投影
(9) if(PSContain(Q2,S)){   // 面包含投影,计算点对象与投影的距离
(10)    dis1 = PPDis(Q1,Q2);
(11)    if(dis1 < dis) dis = dis1;
    }
(12) for each segment Seg of S{   // 计算点到面的每条线段的距离,取最小值
(13)    dis1 = SegSegDis(P1P2, Seg);
(14)    if(dis1 < dis) dis = dis1;
    }
(15) return dis;
```

### 2.5 面 - 面距离计算方法

空间面对象 $S_1$ 与面对象 $S_2$ 的距离可以通过计算面对象 $S_1$ 与构成面对象 $S_2$ 的每条线段的距离以及面对象 $S_2$ 与构成面对象 $S_1$ 的每条线段的距离,取最小值.

### 2.6 相邻关系判断算法 *CRAR*

**算法 5** 三维空间中面对象相邻关系判断算法 CRAR

**输入:** 空间对象集 $D$,距离阈值 $D_{\max}$.

**输出:** $D$ 中各个面对象及其相邻的空间对象.

**步骤:**

```
(1) Point-line-region = GetPointLineRegionObjects(D);   // 获取点线面集合 Point-line-region
(2) Region = GetRegionObjects(D);   // 获取面集合 Region
(3) for each object r of Region do
(4) { Adjacent[r] = NULL;
(5) for each object i of Point-line-region do
(6) { dis = ComputeDistance(r,i);   // 求面 r 与其他对象 i 的距离
(7) if(dis <= Dmax&&dis > 0)   // 满足阈值条件
(8) Adjacent[r] = Adjacent[r] ∪ {s};   // 加入相邻关系集合
        }
    }
```

### 2.7 相邻关系判断算法 *CRAR-DF*

给定空间对象 $R$ 和 $S$. 由于空间对象的 *AABB* 的边至少包含了该对象的一个点,因此 $\mathrm{DIST}(R,S) \geqslant \mathrm{MINDIST}(AABB_R, AABB_S)$. 对于 $\mathrm{MINDIST}(AABB_R, AABB_S) > D_{\max}$,显然,两者不相邻. *AABB* 间的距离计算较为简单,因此可用来提高空间对象相邻关系的计算效率. 判断两个空间对象是否相邻时,可以先计算它们的 *AABB* 的最小距离是否大于 $D_{\max}$,如果大于,则这两个对象一定不相邻,否则精确计算它们的距离再进行判断.

## 3 实验结果与分析

在 Intel Pentium IV 2.93 GHz,内存 512 MB 的微机上,用VC++ 实现了本文提出的三维空间中面对象相邻关系的判断算法. 实验采用合成数据源(如图4 所示),其中五角星代表学校,小三角形代表宾馆,圆点代表商店,虚线代表河流,实线代表公路,粗线代表铁路,三角形代表湖泊,四边形代表住宅区. 根据本文提出的算法,可以计算出住宅区 *S*1 与一所学校、两个商店、一条公路以及一条河流相邻; *S*2 与一所学校、两个

商店、一条公路、两条河流以及一条铁路相邻; $S3$ 与一所学校、两个商店、一家宾馆、一条公路相邻.

图 5 给出了算法 CRAR 和 CRAR-DF 的运行效率比较,可以看出,采用距离函数进行快速判断后,CRAR-DF 的执行时间比 CRAR 少,而且随着空间对象个数的增加,优势更加明显.

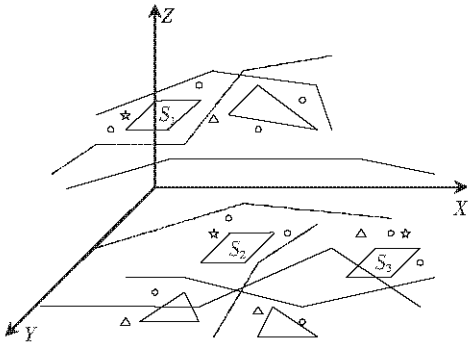


图 4　实验数据集

Fig.4　Experimental data set

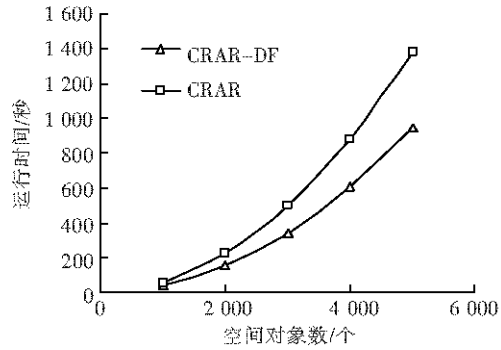


图 5　算法 CRAR 和 CRAR-DF 的运行效率

Fig.5　Runtime of algorithm CRAR and CRAR-DF

## 4　结语

本文提出了三维空间中面对象相邻关系的判断算法 CRAR. 此外,在 CRAR 的基础上利用距离函数进行相邻关系的快速判断,提出了算法 CRAR-DF. 实验结果表明上述两个算法均是有效的. 除了距离函数,还可以通过建立邻近索引等方式进一步提高算法的效率. 此外,三维空间还包含体对象,关于体对象的拓扑关系计算需要深入研究.

[参考文献] (References)


[1] 陈军,邬伦. 数字中国地理空间基础框架 [M]. 北京: 科学出版社,2003: 67-74.
Chen Jun, Wu Lun. Geo-spatial Framework of Digital China [M]. Beijing: Science Press, 2003: 67-74. (in Chinese)

[2] 邓敏,刘文宝,黄杏元,等. 空间目标的拓扑关系及其 GIS 应用分析 [J]. 中国图象图形学报,2006,11(12): 1 744-1 749.
Deng Min, Liu Wenbao, Huang Xingyuan, et al. Modeling topological relations of spatial objects and its applications in GIS [J]. Journal of Image and Graphics, 2006, 11(12): 1 744-1 749. (in Chinese)

[3] 吴立新,史文中,Christopher Gold. 3D GIS 与 3D GMS 中的空间构模技术 [J]. 地理与地理信息科学,2003,19(1): 5-11.
Wu Lixin, Shi Wenzhong, Christopher Gold. Spatial modeling technologies for 3D GIS and 3D GMS [J]. Geography and Geo-Information Science, 2003, 19(1): 5-11. (in Chinese)

[4] Han J, Kamber M. Data Mining: Concepts and Technique [M]. San Fransisco: Morgan Kaufmann Press, 2000.

[5] Yang Na, Ji Genlin. A spatial lines clustering algorithm based on adjacent relations for GML data [C] // Proceedings of 2009 International Conference on Information Engineering and Computer Science. China: Wuhan, 2009: 3 593-3 596.

[6] 田洪军,闫浩文,王丹英,等. 空间关系中两相邻实体间最近距离算法研究 [J]. 测绘科学,2008,33(1): 200-202.
Tian Hongjun, Yan Haowen, Wang Danying, et al. Research on algorithms for nearest distance of two adjacent objects in spatial relation [J]. Science of Surveying and Mapping, 2008, 33(1): 200-202. (in Chinese)

[7] 郭薇,詹平,郭菁. 面向地理信息系统的三维空间数据模型 [J]. 江西科学,1999,17(2): 77-83.
Guo Wei, Zhan Ping, Guo Jing. 3D spatial data modeling for geographic information system [J]. Jiangxi Science, 1999, 17(2): 77-83. (in Chinese)

[8] Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries [C] // Proceedings of the ACM SIGMOD International Conference on the Management of Data. CA: San Jose, 1995: 71-79.


[责任编辑: 严海琳]