文章编号: 1001-2486(2002)01-0071-06

# RSDD 和基于 RSDD 对象的重构算法*

朱铁稳, 肖予钦, 景 宁
(国防科技大学电子科学与工程学院, 湖南 长沙 410073)

摘要: 在 GIS 应用中, 需要处理大量空间数据, 因而所设计的空间数据库系统必须附加存储、检索、处理和查询空间数据的功能。但是用空间数据库来管理空间数据又存在很多困难, 如数字健壮性和拓扑正确性问题。为了解决这个问题, 引入了均匀空间离散域 RSDD, 均匀空间离散域基本对象 RPO 和均匀空间离散域对象 RO 概念, 定义了基于 RSDD 的空间数据类型及其操作。由于计算机系统字长的有限性, 它所处理的数据精度是有限的, 因此所提出的方法尤其适用于在有限精度情况下对几何算法的正确实现。

关键词: RSDD; 空间数据类型, 空间数据库; GIS
中图分类号: TN919 文献标识码: A

# RSDD and Restructure Algorithms for ROs

ZHU Tie-wen, XIAO Yu-qin, JING Ning
(College of Electronic Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

**Abstract:** In the application of GIS, there is a need to manage spatial data. For this purpose spatial database systems are designed to be full-fledged database systems with additional capabilities for storing, retrieving, manipulating, and querying spatial data. But there are many difficulties to manage spatial data using spatial databases, such as numerical robustness and topological correctness. To solve these problems, we introduce some new concepts about Regularly Spatial Discrete Domains (RSDD), RSDD_based Primary Object (RPO), RSDD_based Object (RO), and defined the spatial data types and their operations over RSDD. As in a computer system, it is feasible only with limited precision, i.e., over a finite, discrete, and homogeneous grid, this strategy is ad hoc suitable for correct finite-precision implementations of geometric algorithms.

**Key words:** RSDD; spatial data type; spatial database; GIS

空间数据库是 GIS 应用的核心, 它处理的数据类型是点、线、面、体。在空间数据库的发展进程中, 空间数据类型的定义和实现是一个最为基本的课题。本文的研究是在欧几里德空间中进行, 空间中的点可以用实数坐标表示。理论计算几何[1,2]对基于欧氏空间的几何模型, 提出了许多有用和有效的算法, 但它们要求具有连续的参数和无限的精度, 这与实际的计算机系统只具有有限精度相矛盾, 因而, 利用这些算法编写有效和可靠的程序是很不容易的[3~5]。

这种矛盾导致了计算几何中的许多问题[6,7], 除数据错误外, 还有拓扑不一致性问题。Forrest[8]通过讨论点在多边形内的判定算法和线段相交算法, 说明了在理论计算几何和应用计算几何之间的这种矛盾, Burton FW[9]等人也同样说明了这种矛盾能够发生在简单的判定点在多边形内的算法中。如果在建模时忽略计算机的有限表示问题, 把这个问题留给空间数据库系统开发人员, 这将不可避免地导致查询处理错误。因此一些学者提出了适应于建模和实现的离散几何基础的概念, 例如, Egenhofer M, Frank A 和 Jackson JP[3]的 simplex 和 simplicial complex 概念, G ting RH 和 Schneider M[10,11]的 realm 概念, 但是它们的研究都是局限在 2 维空间, 而本文将研究范围扩展到 3 维空间。

* 收稿日期: 2001-09-07
基本项目: 教育部优秀年轻教师基金资助
作者简介: 朱铁稳 (1962—), 男, 副教授, 博士。

# 1 均匀空间离散域与均匀空间离散域基本对象

我们首先定义均匀空间离散域 RSDD 和均匀空间离散域基本对象 RPO (RPO-point, RPO-plane 和 RPO-solid), 以及关于它们的一些操作。所有这些定义都是为了能够得到可以直接实现的无错误整数算法。设 $N = \{0, 1, \ldots, n-1\}$, $n$ 是整数, RSDD 定义为 $N \times N \times N$ 中点的集合。称 RSDD 中的点为 RPO-point, 即 RPO-point 是一个元组 $(x, y, z) \in N \times N \times N$, $x$, $y$, $z$ 分别是三个坐标方向上的坐标; RPO-line 是端点位于 RSDD 中的线段; RPO-plane 是由共面的 RPO-line 组成的封闭图形。RPO-solid 是由 RPO-plane 所组成的封闭立体, 见图 1 所示。

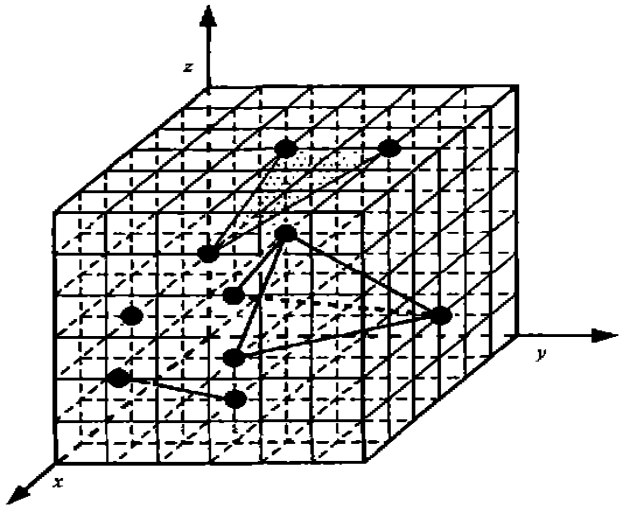


图 1 均匀空间离散域与均匀空间离散域基本对象

Fig 1 Regularly spatial discrete domains and RSDD_based primary object

# 2 均匀空间离散域对象

## 2.1 均匀空间离散域对象的定义

对于给定 $N$, 设 $PO_N$ 是所有 RPO-point 的集合, $LI_N$ 是所有 RPO-line 的集合, $PL_N$ 和 $SO_N$ 分别是所有 RPO-plane 和 RPO-solid 集合。均匀空间离散域对象定义为集合 $RO = PO_{sub} \cup LI_{sub} \cup PL_{sub} \cup SO_{sub}$, 并且满足以下条件:

(1) $PO_{sub} \subseteq PO_N$, $LI_{sub} \subseteq LI_N$, $PL_{sub} \subseteq PL_N$, $SO_{sub} \subseteq SO_N$;

(2) $\forall l \in LI_{sub}$: $l = (p_1, p_2)$, $p_1 \in PO_{sub} \wedge p_2 \in PO_{sub}$;

(3) $\forall p \in PO_{sub}$, $l \in LI_{sub}$: $\neg (p \text{ in } l)$;

(4) $\forall l \in LI_{sub}$, $s \in PL_{sub}$: when $\neg (l \text{ in } s) \wedge$ have intersection, (intersection of $l$ and $s$) $\in PO_{sub}$;

(5) $\forall l \in LI_{sub}$, $v \in SO_{sub}$: when $\neg (l \text{ in } v) \wedge$ have intersection, (intersection of $l$ and $v$) $\in PO_{sub}$;

(6) $\forall s \in PL_{sub}$, $v \in SO_{sub}$: when $\neg (s \text{ in } v) \wedge$ have intersection, (intersection of $s$ and $v$) $\in PO_{sub} \cup LI_{sub}$;

(7) $\forall l_1, l_2 \in LI_{sub}$: $\neg (l_1 = l_2) \wedge \neg (l_1$ and $l_2$ intersect) $\wedge \neg (l_1$ and $l_2$ overlap);

(8) $\forall s_1, s_2 \in PL_{sub}$: $\neg (s_1 = s_2) \wedge \neg (s_1$ and $s_2$ intersect) $\wedge \neg (s_1$ and $s_2$ overlap);

(9) $\forall v_1, v_2 \in SO_{sub}$: $\neg (v_1 = v_2) \wedge \neg (v_1$ and $v_2$ meet in plane) $\wedge \neg (v_1$ and $v_2$ overlap).

根据以上定义, 一个均匀空间离散域对象实际上就是一些顶点位于 RSDD 栅格点的均匀空间离散域基本对象的集合。其中线的两个端点位于 RSDD 中, 并且除了端点以外再无其它的 RPO-point。对于 RPO-line 和 RPO-plane, 它们要么不相交, 要么交于 RPO-point。对于 RPO-line 和 RPO-solid, 它们要么

不相交，要么交于 RPO-point。对于 RPO-plane 和 RPO-solid，它们要么不相交，要么交于 RPO-point 或 RPO-line。对于任意两条 RPO-line 或两个 RPO-plane，它们既互不相等，也不相交和重叠。对于任意两个 RPO-solid，它们既互不相等，也不相交于一个平面和重叠。

**2.2** 对均匀空间离散域基本对象相交情形的调整算法

根据定义，均匀空间离散域基本对象端点或顶点的坐标都是整数。因为在计算机内存中对整数的计算和表示都不会产生错误，所以用计算机管理均匀空间离散域基本对象是非常合适的。这种方法保证了相关对象的几何一致性和空间数据类型的计算封闭性。但是现实应用中的几何数据并不排除相交的情况，所以必须考虑两个几何体的相交，也就是说当两个几何体相交时，需要重构几何体以满足我们的定义。基本的重构过程可以分为以下三种：重画一条线、重构一个面和重构一个体。当进行重构时，往往需要将一个点移动到与它最邻近的 RPO-point，例如两条 RPO-line 的交点通常不在 RSDD 的栅格点上，需要将它移到 RSDD 的栅格点上。

2.2.1 将一个点移动到与其最近的 RPO-point 的算法

设该点的坐标为 $(x, y, z)$，如果 $x, y, z$ 都是整数，即该点为 RPO-point，则不需要进行移动。如果只有 $x$ 和 $y$ 是整数，则该点必位于端点为 $(x, y, [z])$ 和 $(x, y, [z]+1)$ 的栅格线上，其中 $[z]$ 是小于或等于 $z$ 的最大整数。计算 $z-[z]$ 和 $[z]+1-z$，如果 $z-[z]<0.5$，移动该点到 $(x, y, [z])$；如果 $[z]+1-z<0.5$，移动该点到 $(x, y, [z]+1)$；如果 $z-[z]=[z]+1-z$，移动该点到 $(x, y, [z])$ 或者 $(x, y, [z]+1)$ 均可。如果只有 $x$ 是整数，则该点必位于顶点为 $(x, [y], [z])$，$(x, [y]+1, [z])$，$(x, [y]+1, [z]+1)$ 和 $(x, [y], [z]+1)$ 的栅格上。分别计算 $(y-[y])^2+(z-[z])^2$，$(y-[y]-1)^2+(z-[z])^2$，$(y-[y]-1)^2+(z-[z]-1)^2$ 和 $(y-[y])^2+(z-[z]-1)^2$。类似地，根据这些值移动该点到值最小的那个顶点。如果 $x, y$ 和 $z$ 都不是整数，则该点必位于顶点为 $([x], [y], [z])$，$([x], [y]+1, [z])$，$([x], [y]+1, [z]+1)$，$([x], [y], [z]+1)$，$([x]+1, [y], [z])$，$([x]+1, [y]+1, [z])$，$([x]+1, [y]+1, [z]+1)$ 和 $([x]+1, [y], [z]+1)$ 的栅格体内。分别计算该点和这些顶点之间的距离，根据这些值移动该点到值最小的那个顶点。

2.2.2 重画一条 RPO-line 的算法

设在 RPO-line $l$ 上添加一个点，则必须重画 $l$ 以满足定义。此时不能只简单地将该点移动到与其最近的 RPO-point，然后将 RPO-point 与 RPO-line $l$ 的端点相连接，因为这样的做法会导致拓扑错误，避免错误的基本思想是将它转换成一个线段链[11]。为了解决这个问题，我们把 Greene 和 Yao 的方法[6]扩充到 3 维空间，不失一般性，假设 RPO-line $l$ 和平面 $XOY$ 之间的角度小于等于 45°，并且在平面 $XOY$ 中 RPO-line $l$ 的投影与轴 $OX$ 之间的角度小于等于 45°。我们定义关于 RPO-line $l$ 的正方形如下：设 RPO-line $l$ 的端点为 $(x_1, y_1, z_1)$ 和 $(x_2, y_2, z_2)$ 且 $x_1<x_2$，对于 $l$ 上的任意一点 $(x, y, z)$，其中 $x \in (x_1, x_2)$ 是整数，正方形 $SQ(x)$ 由点 $(x, [y], [z])$，$(x, [y]+1, [z])$，$(x, [y]+1, [z]+1)$ 和 $(x, [y], [z]+1)$ 构成。

如果所增加的点是 RPO-line 的端点，什么都不需要做。如果所加的点在 RPO-line 上，需要将该点移动到与其最近的 RPO-point。连接该点和 RPO-line 的两个端点后，从 $x_1$ 到 $x_2$ 依次检验每个整数 $x$，看所画线段是否仍在正方形 $SQ(x)$ 中。如果 $(x, y, z)$ 是第一个不在 $SQ(x)$ 中的点，计算该点和 $SQ(x)$ 各顶点之间的距离，根据这些值移动该点到值最小的那个顶点，然后重画这些线。重复此过程，直到所有线段都在正方形 $SQ(x)$ 中为止。

2.2.3 重构一个 RPO-plane 的算法

设在 RPO-plane 上添加一个点，则需重构平面以满足定义。假设 RPO-plane 的顶点集为 $V=\{v_1, v_2, \cdots, v_n\}$，如果所加点是 $V$ 中的点，什么都不做。如果所加点位于 RPO-plane 的边界上，它必位于某条 RPO-line 上，不妨假设这条 RPO-line 为 $L$。利用重画一条 RPO-line 的算法重画 $L$，则 $L$ 变换成一系列 RPO-line $L_1, L_2, \cdots, L_m$，并且它们的端点为 $u_1, u_2, \cdots, u_{m+1}$，其中 $u_1$ 和 $u_{m+1}$ 是 $L$ 的端点。检验 $u_2$ 是否在 RPO-plane 所在的平面上，如果是的话，将三角形 $u_1u_2u_{m+1}$ 添加到 RPO-plane 上并进一步检验 $u_3$；如果不是的话，将三角形 $u_1u_2u_{m+1}$ 作为另一个 RPO-plane 并检验 $u_3$ 是否在这个新的 RPO-

plane 所在的平面上。如果 $u_3$ 在这个新的 RPO-plane 所在的平面上，将三角形 $u_2u_3u_{m+1}$ 添加到新的 RPO-plane 上并进一步检验 $u_4$；如果不在这个平面上，按照前面的方法同样处理，直至全部检验为止。如果所加点在 RPO-plane 内部，将该点移动到与其最近的 RPO-point $P$。如果 $P$ 在 RPO-plane 上，将 P 添加到 RPO-plane 上；否则分别连接 $P$ 和 $V$ 中各顶点，构成三角形 $Pv_1v_2$，$Pv_2v_3$，…，$Pv_nv_1$。

2.2.4 重构一个RPO-solid 的算法

设在 RPO-solid 上添加一个点，则必须重构此体以满足定义。如果所加点在 RPO-solid 内部，将该点移动到与其最近的 RPO-point 并将此 RPO-point 点添加到 RPO-solid 上。如果所加点位于体的边界面上，利用重构 RPO-plane 的算法重构该边界面，将重构所得的 RPO-plane 作为 RPO-solid 的新的边界面，重构 RPO-solid。

## 3 对均匀空间离散域对象的操作

显然对均匀空间离散域对象 RO 的基本操作包括插入一个均匀空间离散域基本对象 RPO 和删除一个均匀空间离散域对象 RO。因为我们所研究的均匀空间离散域对象不只是抽象实体，而且还与在计算机中的具体实现相关，所以操作过程比较复杂。因而必须讨论插入一个 RPO 和删除一个 RO 的有关算法。

**3.1 插入一个 RPO-point 的算法：InsertRPO-point**

当在 RO 中插入一个 RPO-point $p$ 时，有以下 3 种情况：

(1) $p$ 已经存在于 RO 中；(2) $p$ 不在 RO 中的任何 RPO 上；(3) $p$ 在 RO 中的某个 RPO 上；此时必须区分 $p$ 是否是 RPO-line 的端点，或者 RPO-plane 和 RPO-solid 的顶点。

步骤 1：验证 $p$ 是否在 RO 中。如果在的话，转到步骤 4。

步骤 2：验证 $p$ 是否在 RO 上。如果不在 RO 中的任何 RPO 上，只需将 $p$ 添加到 RO 中。

步骤 3：如果 $p$ 在 RO 中的某个 RPO 上，验证 $p$ 在 RO 中的哪个 RPO 上，是 RPO-line、RPO-plane 还是 RPO-solid。步骤 31：如果 $p$ 在 RPO-line 上并且是它的某个端点，转到步骤 4。否则，将 $p$ 作为添加点，重画 RPO-line。步骤 32：如果 $p$ 在 RPO-plane 上并且是在它的边界上，转到步骤 31。如果 $p$ 在 RPO-plane 内部，只需添加该点。步骤 33：如果 $p$ 在 RPO-solid 上并且是在其边界面上，转到步骤 32。否则，只需添加该点。

步骤 4：结束。

**3.2 插入一条 RPO-line 的算法：InsertRPO-line**

当在 RO 中插入一条 RPO-line $l$ 时，有以下 2 种情况：

(1) RO 中是否已经存在 $l$；(2) 需要验证 $l$ 和 RO 中 RPO 的相交情况。

步骤 1：验证 $l$ 是否在 RO 中。如果在的话，转到步骤 4。

步骤 2：验证 $l$ 是否与 RO 中的 RPO 相交。如果不交的话，只需将 $l$ 添加到 RO 中。

步骤 3：如果 $l$ 与 RO 中的某个 RPO 相交的话，验证 $l$ 与 RO 中的哪个 RPO 相交，是 RPO-point、RPO-line、RPO-plane 还是 RPO-solid。步骤 31：如果 $l$ 与 RO 中的 RPO-point 相交，并且交点是这个 RPO-line 的端点，转到步骤 4。否则，将该交点移动到与其最近的 RPO-point（如果交点不是 RPO-point，这个移动过程是需要的。移动一个点到它的最近的栅格点的算法见前面所述），并将此 RPO-point 作为添加点，重画 RPO-line。步骤 32：如果 $l$ 与 RO 中的 RPO-line 相交，并且交点是两条 RPO-line 的共同端点，转到步骤 4。如果交点只是其中一条 RPO-line 的端点，将该交点作为添加点，重画交点不是端点的那条 RPO-line。如果交点不是两条 RPO-line 的任何端点，将该交点移动到与其最近的 RPO-point，重画两条 RPO-line。步骤 33：如果 $l$ 与 RO 中的 RPO-plane 相交，并且交点在 RPO-plane 的边界上，转到步骤 32。如果交点在 RPO-plane 的内部，并且该交点是 RPO-point，转到步骤 4。否则将该交点移动到与其最近的 RPO-point，连接此 RPO-point 和 RPO-plane 的顶点，重构 RPO-plane。如果 RPO-line 与 RPO-plane 共面，则 RPO-line 必与 RPO-plane 的边界交于一点。如果此点是 RPO-point，转步骤 4；否则将该交点移动到与其最近的 RPO-point，重画 RPO-line 和包含该交点的边界。步骤 34：如果 $l$

与 RO 中的 RPO-solid 相交, 它们必交于一点或一条线。只需考虑 $l$ 与 RPO-solid 的边界面 RPO-plane 相交的情况, 因而转到步骤 33。

步骤 4: 结束。

### 3.3 插入一个 RPO-plane 的算法: InsertRPO-plane

设插入的 RPO-plane 为 $f$, 则算法 InsertRPO-plane 的步骤为:

步骤 1: 验证 $f$ 是否在 RO 中。如果在的话, 转到步骤 4。

步骤 2: 验证 $f$ 是否与 RO 中的某个 RPO 相交。如果不交的话, 只需将 $f$ 添加到 RO 中。

步骤 3: 如果 $f$ 与 RO 中的某个 RPO 相交, 验证 $f$ 与 RO 中的哪个 RPO 相交, 是 RPO-point、RPO-line、RPO-plane 还是 RPO-solid。步骤 31: 如果 $f$ 与 RO 中的 RPO-point 相交, 则此 RPO-point 必位于 RPO-plane 的边界或内部。如果在边界上, 则此 RPO-point 必在某条 RPO-line 上。若 RPO-point 是 RPO-line 的端点, 转到步骤 4。否则, 将该 RPO-point 作为添加点重画 RPO-line。如果在内部, 只需添加 $f$。步骤 32: 如果 $f$ 与 RO 中的 RPO-line 相交 (不共面), 并且交于 RPO-plane 的边界上一点, 只需添加 $f$。若交点在 RPO-plane 内部且不是 RPO-point 的话, 将该交点移动到与其最近的 RPO-point, 重画 RPO-line。如果 $f$ 与 RPO-line 共面, 它们必交于边界上一点。将该点作为添加点重画 RPO-line 和 RPO-plane 的边界, 并添加 $f$。步骤 33: 如果 $f$ 与 RO 中的 RPO-plane 相交并且它们共面, 若交于一个点或者一条线, 转到步骤 4。若它们相互重叠, 考虑两个 RPO-plane 边界线 RPO-line 的交点, 将该交点移动到与其最近的 RPO-point, 重画两个 RPO-plane 的边界线 RPO-line, 并且删除一个 RPO-plane 在另一个 RPO-plane 中的边界部分, 重构 $f$。如果 $f$ 与 RO 中的 RPO-plane 相交并且它们不共面, 若交于一点, 转到步骤 4。若交于一条线, 将该交线的两个端点移动到与其最近的 RPO-point, 添加这两个 RPO-point 作为两个 RPO-plane 的顶点, 重构两个 RPO-plane。步骤 34: 如果 $f$ 与 RO 中的 RPO-solid 相交, 若它们交于一点, 转到步骤 4。若交于一条线, 则该线必位于 RPO-solid 边界面上, 并且其端点是 RPO-point, 添加两个端点作为 RPO-solid 的顶点, 重构此 RPO-solid。若交于一个面, 则相交部分是一个多边形, 将此多边形的顶点分别移动到与其最近的 RPO-point, 添加这些 RPO-point 作为 RPO-solid 和 RPO-plane 的顶点, 重构 RPO-solid 和 RPO-plane。

步骤 4: 结束。

### 3.4 插入一个 RPO-solid 的算法: InsertRPO-solid

设插入的 RPO-solid 为 $s$, 则算法 InsertRPO-solid 的步骤为:

步骤 1: 验证 $s$ 是否在 RO 中。如果是的话, 转到步骤 4。

步骤 2: 验证 $s$ 是否与 RO 中的某个 RPO 相交。如果不交的话, 只需将 $s$ 添加到 RO 中。

步骤 3: 如果 $s$ 与 RO 中的某个 RPO 相交的话, 验证 $s$ 与 RO 中的哪个部分相交, 是 RPO-point、RPO-line、RPO-plane 还是 RPO-solid。步骤 31: 如果 $s$ 与 RO 中的 RPO-point 相交, 并且交点是 RPO-solid 的顶点, 转到步骤 4。若交点在 RPO-solid 的边界线上, 将该 RPO-point 作为添加点, 重画边界线。若交点在 RPO-solid 或其边界面内部, 只需添加 $s$。步骤 32: 如果 $s$ 与 RO 中的 RPO-line 相交, 它们必交于一个点或一条线。若交于一点, 则交点必在 RPO-solid 的边界面上, 且该交点是 RPO-point, 转到步骤 4。若交于一条线, 并且交线在 RPO-solid 内部, 则该交线自身就是 RPO-line, 只需添加 $s$; 否则, 考虑 RPO-solid 边界面与 RPO-line 的交点, 将该交点移动到与其最近的 RPO-point, 以此 RPO-point 作为添加点, 重画 RPO-line 并重构 RPO-solid 的边界面。步骤 33: 如果 $s$ 与 RO 中的 RPO-plane 相交, 它们必交于一个点、一条线或一个面。若交于一点, 则交点必是 RPO-solid 或 RPO-plane 的顶点, 分别将此 RPO-point 作为添加点, 重构 RPO-plane 或 RPO-solid (如果需要)。若交于一条线, 则交线必位于 RPO-solid 的边界面和 RPO-plane 上, 将交线的两个端点移动到与其最近的 RPO-point 并将这两个 RPO-point 作为 RPO-solid 和 RPO-plane 的顶点, 重构 RPO-plane 和 RPO-solid。若交于一个面, 则相交部分是一个多边形, 将此多边形的顶点分别移动到与其最近的 RPO-point, 添加这些 RPO-point 作为 RPO-solid 和 RPO-plane 的顶点, 重构 RPO-solid 和 RPO-plane。步骤 34: 如果 $s$ 与 RO 中的 RPO-solid 相交, 它们必交于一个点、一条线、一个面或一个体。此时, 将交点移动到与其最近的 RPO-point, 添加这些 RPO-

point 作为 RPO-solid 的顶点，重构两个 RPO-solid，即计算它们的最大包围多面体。

步骤 4：结束。

从以上讨论可知，一个均匀空间离散域对象包括若干 RPO-point、RPO-line、RPO-plane 和 RPO-solid。因此，均匀空间离散域上的空间数据类型是 RPO-point、RPO-line、RPO-plane 和 RPO-solid。

## 4 结束语

本文中，我们定义了基于 RSDD 的空间数据类型和它的基本操作算法，数据类型的特点是空间对象的点和顶点位于均匀空间离散域的栅格点上，坐标是整数，优点是便于利用数据库管理系统和计算机系统来处理这种数据，目的在于能够利用 DBMS 处理 GIS 中的 3 维空间数据，RSDD 是达到这个目的的一个非常重要的基础。值得指出的是，GIS 中对空间数据的处理，还有许多工作要做，如利用 DBMS 管理空间数据的实现机制研究，对空间数据进行查询、检索的算法研究与实现，在数据库中对空间数据进行插入、删除、更新等维护操作等等，由于涉及的是 3 维空间数据，这些问题都是相当困难和复杂的。

## 参考文献:

[1] Forrest A R. Computational Geometry and Software Engineering: Towards a Geometric Computing Environment State-of-the Art in computer Graphics [M]. Techniques for Computer Graphics, Springer Verlag, 1987, 23-37.

[2] Yao F F. Computational Geometry Algorithms and Complexity [M]. Elsevier Science Publishers: Handbook of Theoretical Computer Science, vol. A, 1992, 343-389.

[3] Egenhofer M J, Frank A, Jackson J P. A Topological Data Model for Spatial Databases [C]. Proc. 1st Int. Symposium on Large Spatial Databases, Santa Barbara, 1989, 271-286.

[4] Egenhofer M J, Herring J. High-Level Spatial Data Structures [M]. Geographical Informations Systems: Overview, Principles and Applications, 1991.

[5] Kriegel H P, Brinkhoff T, Schneider R. The Combination of Spatial Access Methods and Computational Geometry in Geographical Database Systems [C]. Proc. of the 2nd Int. Symp On Advances in Spatial Databases, 1991, 5-21.

[6] Greene D, Yao F. Finite-Resolution Computational Geometry [C]. Proc. 27th IEEE Symp on Foundations of Computer Science, 1986, 143-152

[7] Franklin W R. Cartographic Errors Symptomatic of Underlying Algebra Problems [C]. Proc. 1st Int. Symposium on Spatial Data Handling, Z rich, 1984, 190-208.

[8] Forrest A R. Computational Geometry in Practice [J]. Fundamental Algorithms for Computer Graphics, Springer Verlag, 1985, 707-723.

[9] Burton F W, Kollias V J, Kollias J G. Consistency in Point-in-Polygon Tests [J]. The Computer Journal, 1984, 27 (4): 375-376

[10] G ting R H, Schneider M. Realms: A Foundation for Spatial Data Types in Database Systems [C]. Proc. 3rd Int. Symposium on Large Spatial Databases, Singapore, 1993, 14-35.

[11] Markus Schneider. Spatial Data Types for Database Systems-Finite Resolution Geometry for Geographic Information System [M]. Springer-Verlag, Berlin Heidelberg, 1997.