点云数据压缩中的边界特征检测

钱锦锋[1)] 陈志杨[2)] 张三元[1)] 叶修梓[1)]
[1)](浙江大学计算机学院,杭州 310027) [2)](浙江工业大学软件学院,杭州 310014)

摘 要 点云数据压缩是逆向工程产品建模中必要的数据预处理手段之一。常见的数据压缩算法未考虑点云边界数据点的保留问题,因此在大比例压缩过程中会出现边界数据丢失的情况,从而破坏了数据的完整性。为此,提出了一种利用点云数据小邻域内点的相邻关系来检测边界特征点的算法。该算法能检测出点云数据的内、外边界特征点,同时对边界上的点进行排序,检测出边界特征点中的过渡点,最后构建点云轮廓的边界多边线。该算法不仅能满足在点云数据压缩过程中检测并保留边界特征点的要求,而且生成的边界多边线也为后面的 3 维模型重建奠定了基础。

关键词 散乱点云 边界点 BSP 树 点云压缩
中图法分类号: TP391 **文献标识码**: A **文章编号**: 1006-8961(2005)02-0164-06

The Detection of Boundary Point of Point Cloud Compression

QIAN Jin-feng[1)], CHEN Zhi-yang[2)], ZHANG San-yuan[1)], YE Xiu-zi[1)]
[1)](*College of Computer Science & Technology, Zhejiang University, Hangzhou* 310027)
[2)](*College of Software Engineering, Zhejiang University of Technology, Hangzhou* 310014)

Abstract The compression of point cloud is one step of the necessary preprocessing in the reverse engineering modeling. The issue of detection and preservation of boundary points, however, has not been usually considered in many point cloud compression algorithms. Then, boundary points could be withdrawn during large-scale compression, and integrality of data could not be guaranteed. In this paper, we provide an algorithm to detect the boundary points. Obviously, if a point is a boundary point, then its circumambient points will distribute only on one side or around a corner. If a point is not a boundary point, its neighboring points will distribute symmetrically. By this way, boundary points will be detected by analyzing the relation of points in a trivial neighboring region. The algorithm will be effective for inner boundary points as well as outers. It sorts the boundary points further, and detects the transitional points in boundary points, at last constructs boundary polygon lines by transitional points. The distance between a boundary point and its neighboring boundary points is used to detect the transitional points. Therefore, the algorithm not only can satisfy preserving the boundary points in a large-scale compression, but also prepare for reversion modeling by boundary polygon lines.

Keywords unordered point cloud, boundary points, BSP tree, point cloud compression

1 引 言

由于测量设备的数字化、自动化、测量精度的不断提高,模型的测量数据呈快速增长趋势。目前一般的激光测量设备可以从产品表面轻易获取数十万,甚至数百万的测量数据。面对如此大量的数据,如何提高数据的利用率是后序产品造型工作面临的严峻问题之一。目前的商业 CAD 系统一般无法直接处理这样的海量数据,一种比较常用的策略是通过专用的逆向工程(reverse engineering, RE)系统首先对数据进行处理,得到基础曲线曲面框架之后再利用商业 CAD 软件的强大造型功能完成产品从测量数据到 CAD 模型的转换[1~3]。即便如此,百万数

基金项目:国家科技部软件重大专项(2003AA4Z1020);国家自然科学基金项目(60273060;60073026)
收稿日期:2004-04-23;**改回日期**:2004-07-19

第一作者简介:钱锦锋(1979~),男,2002 年于西安交通大学获机械工程及自动化专业学士学位,现于浙江大学计算机科学与技术学院攻读计算机应用专业硕士学位。主要研究方向为计算机图形学和 CAD。E-mail:guy.feng@163.com

量级的测量数据无论对逆向工程系统本身还是对计算机存储系统都提出了严峻的考验。显然,不加处理地直接应用这些数据是不合理的也是不现实的(100 万的 3 维数据点在计算机中大约要占用 100M 的存储空间),如此巨量的测量数据不仅加大了系统的负荷,而且大大降低了后续处理的效率。因此点云数据的预处理,特别是点云数据的压缩已成为逆向工程产品造型一个必不可少的预处理过程。点云数据的预处理,确切地说是点云的压缩,对于提高网格模型生成、特征提取效率以及 3 维 CAD 模型的重建都将具有重要意义。

由于实际物体的复杂表面以及不同测量设备的差异,在点云数据压缩处理中的散乱点云数据千差万别,包括各种封闭或不封闭的点云数据。Pauly[4] 使用的点云数据压缩方法基本上是针对闭空间(没有边界的)点云数据,使用重复去除的方法来达到点云压缩。该方法存在的一个重要问题是:对于一些不封闭有边界的点云数据而言,如果边界上的点与它周围点属性特征相似(例如它们实际上在同一平面内),则这些边界数据点就有可能被去除掉,这样压缩结果就丧失了边界,使压缩数据失去了真实的意义。因此在点云数据压缩时,有必要知道哪些点是边界特征点,压缩的时候边界特征点就要想办法保留,以保证点云数据压缩结果具有正确的轮廓特征。

本文提出了一种点云数据压缩中边界特征点检测的方法,该方法通过邻域数据点间关系,实现了保留边界数据的点云数据压缩,从而避免了因数据压缩带来的数据失真现象。

需要指出的是,本文算法研究的对象是开区域上的点云数据,对于封闭区域上的点云数据由于不存在"保边界"问题,因此不在研究范围之内。

2　算法概述

边界特征点检测需要利用数据点的拓扑关系以及数据点法矢等信息构造最小二乘平面,然后根据数据点在平面上的投影点之间的关系进行判断。算法包括以下几个步骤:

(1) 数据点拓扑关系计算　这里所指的拓扑关系是点的邻域关系(即确定每个数据点在给定区域内的相邻数据点,以下简称为 *K*-Nearest Points)。构建拓扑关系的方法有很多,例如 α-shape、三角剖分、最小生成树[5] 等等,本文利用构建空间二元划分(binary space partition, BSP)树[4] 的方法来计算 *K*-Nearest Points。

(2) 数据点法矢计算　利用上一步得到的 *K*-Nearest Points数据点构造最二小乘平面,计算该平面的法矢,并用该法矢作为数据点 *P* 的法矢。

(3) 数据点投影并判断边界点　将数据点 *P* 的 *K*-Nearest Points 投影到最小二乘平面上,并根据投影点的均匀性来判断该点是否为边界特征点。若该点被标记为边界特征点,则点云数据压缩时可以用加权方法将该点保留下来。

(4) 边界多边线的生成　以上检测出的边界特征点是无序的,为了后续重构曲面的需要,对无序的边界点数据进行排序得到多边线。排序采用双向最近点搜索方法,即按照数据点的最近距离从两个方向进行排序从而得到边界的多边线形式表示。

图 1 所示是整个算法的流程。

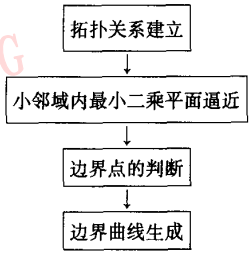


图 1　边界点的提取算法

Fig. 1　Extraction of boundary point algorithm

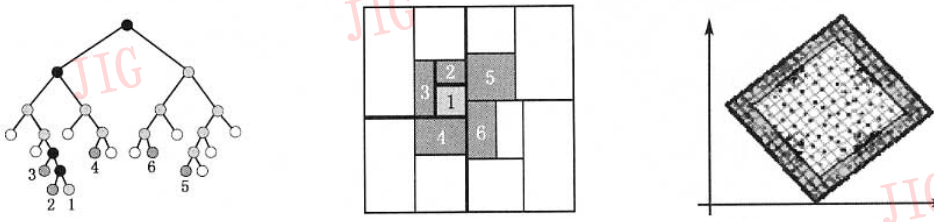
3　*K*-Nearest Points 搜索算法

K-Nearest Points 搜索问题是逆向工程中点云数据预处理中常见的算法之一。*K*-Nearest Points 最直观的方法即在整个点云中遍历查找每个点,通过计算点距离确定某点是否为点 *P* 的 *K*-Nearest Points。显而易见这是一种效率很低的方法,在处理大规模测量数据中采用这种方法是不现实的。其他算法如利用平面点集最近点对算法[6] 扩展到 3 维空间的最近点对来计算 *K*-Nearest Points 最近点。以上这些方法的关键问题是如何确定查找 *K*-Nearest Points 时的搜索范围以及尽可能最小化该搜索范围以节约计算时间,提高算法效率。就空间散乱点云而言,搜索范围的常用确定方法是空间划分——对空间点云数据构建 BSP 树,从而限定点 *K* 最近点的搜索范围。利用这种方法对大规模的点云数据求

K-Nearest Points能极大提高处理的效率。

BSP 树中空间的大小用包围盒来表示，根节点是容纳所有点的一个大包围盒(图 2(a))。父节点不断地划分生成新的叶子节点，同时父节点中的点云数据也被划分到子节点中去。节点是否划分则取决于节点中所含点云数据的点数，节点中的点数达到一个阈值，节点不再划分。其中阈值决定于整个点云数据的大小。需要指出的是节点在空间上划分的方向可以有 3 个不同的方向而且切分平面的位置可以不定，而平面只有两个划分方向(如图 2(b))。

用 BSP 树来组织点后，计算 *K* 最近点还有一个冗余问题。如图 2(b)平面中的情况而言，计算框 1 中某一点的 *K* 最近点，不仅要计算该点与框 1 中其他剩余点的距离，而且还要计算与框 1 相邻周围框(如框 2、框 3、框 6 等)中点的距离，也就是说在计算时要考虑一定的冗余点(如图 2(c)所示灰色地带中的点)，冗余点的多少则取决于在各个方向所加的冗余点是否能够保证正确的计算 *K* 最近点(图 3 为 *K* 最近点计算不准确时检测到的边界点)，在文献[4]中冗余点数取决于包围盒的大小，这里计算 *K* 最近点时冗余点数是保证每个方向(26 相邻方向如图 4)都有冗余点。



(a) BSP 树　(b) 空间 BSP 树中邻接点　(c) 冗余点关系投影到 2 维的示意图

图 2

Fig. 2

图 3　*K* 最近点计算不准确时错误的边界

Fig. 3　Error boundary points when *K* neighboring points is incorrrect

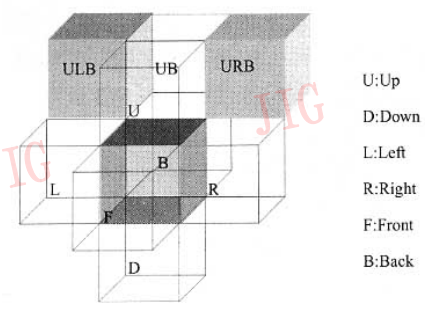


图 4　空间单元相邻关系

Fig. 4　Relation of parts of BSP

相对于平面来讲，空间情况下计算 *K* 最近点就更为复杂。在 BSP 树(图 2(a))中，每个叶子节点(图 4 中正中间的正方体)内某点的 *K* 最近点有可能存在于本身包围盒或者附近的 26 个包围盒中。而且构建完 BSP 树后叶子节点(如图 2(a)中 1 号)中某一点的 *K* 最近点可以分布在 BSP 树各个子树中而不仅仅是相同父节点下的兄弟结点(图 2(b)为投影到 2 维的示意图)，因此在设计 BSP 树节点的结构时，需要用特殊的字段来记录节点之间的相邻关系。另一个需要注意的是在构建完 BSP 树以后，每个叶子节点相邻的周围包围盒数不一定是 26 个，有可能多，也有可能少(如图 4 中 ULB、UB、URB 可以指向 1 个到 3 个包围盒)。BSP 树节点的结构以及确定 *K* 最近点搜索范围的算法如下：

(1) 检查每个叶子节点的 6 个方向(U、D、L、R、F、B)；

(2) 得到每一方向的节点指针和指针所指向的节点被切分时的切分平面及位置；

(3) 判断指针所指的节点是否为叶子节点，是转步骤 4，不是转步骤 6；

(4) 判断指针所指的节点与叶子节点在切分平面上的投影是否相邻，是转步骤 5，不是转步骤 2；

(5) 保存指针到叶子节点的相邻包围盒的数组中;

(6) 用指针所指向节点的子节点重新给指针赋值,转步骤 3。

4 检查边界特征点算法

对于点云数据,可以直观地认为:数据点 P 的 K-Nearest Points 的分布如果偏向一侧,则可以认为点 P 为边界特征点;反之,如果数据点围绕 P 均匀分布,则可以认为 P 是内部点。基于这种思想可以利用数据点及其周围点的分布均匀性来判断边界点。此处均匀性度量标准可以采用角度标准差。角度标准差的计算过程如下:

取点 P 在 K-Nearest Points 中最近点 Q_i 做有向线段$\overrightarrow{PQ_i}$(如图 5 所示),以$\overrightarrow{PQ_i}$ 为基准,计算其余点 $\overrightarrow{PQ_j}$ 与$\overrightarrow{PQ_i}$ 的夹角 α,得到一个角度序列 $\boldsymbol{S}=(\alpha_1,\alpha_2,\cdots,\alpha_n)$。对角度序列 $\boldsymbol{S}$ 按照升序排序,得到新角度序列 $\boldsymbol{S}'=(\alpha_1',\alpha_2',\cdots,\alpha_n')$。定义差序列 $\boldsymbol{L}$,

$$L_i=\begin{cases}\alpha_{i+1}'-\alpha_i' & 1\leqslant i<n\\ \alpha_i'-\alpha_1' & i=n\end{cases}$$

计算差序列 $\boldsymbol{L}$ 的标准差

$$E=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(L_i-\bar{L})^2}$$

其中,$\bar{L}=\frac{1}{n}\sum_{i=1}^{n}L_i$。

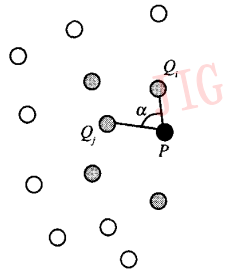


图 5　边界点(实心点为 K-Nearest Points)

Fig. 5　Boundary points(solid points is K neighboring points)

若标准差 E 大于设定的阈值,可以判定 K 最近点分布不均匀即认为此点为边界特征点,反之则可以判断该点为内部点。阈值的大小可以根据点云数据边界的空间复杂度而定,由于实际物体边界的复杂程度不同,这个值应该根据实际情况进行调整。

边界特征点检测的算法流程如下:

(1) 利用 K 最近点逼近最小二乘平面,平面法矢即表示为该点的法矢;

(2) 把 K 最近点投影到平面上,得到一无序点集;

(3) 对无序点集依夹角的大小排序;

(4) 排序完成后计算夹角标准差,当标准差值超过设定阈值时,该点即判为边界特征点。

5 特征点排序

在检测边界特征点后,应对无序的特征点进行排序,将无序的点排列成多边线的形式,这样才能对后续的实体造型具有实际的意义。这里采用搜索最近点的方法来生成多边线,首先在边界特征点中任取一点作为多边线的起始点 P_s,取它的最近点作为多边线的终点 P_e,然后多边线沿两端往外长。任取多边线的一端,比如 P_s 端,在剩余的点中取 P_s 的最近点 P,计算点 P 到 P_s 的距离 d_s 和到 P_e 的距离 d_e,比较 d_e 和 d_s 的大小,若 $d_s<d_e$,则把点 P 插入到 P_s 点前并作为新的起始点;否则往另一段 P_e 生长,同样取 P_e 的最近点 P 计算距离 d_s、d_e,若 $d_e<d_s$,则点 P 插入到 P_e 点前面作为新的终点,否则往 P_s 端生长。如图 6 所示,如此不断地生长,无序的点可排列成一条有序的多边线。

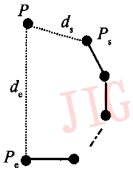


图 6　特征点排序

Fig. 6　Sorting feature points

实际上由于边界特征点构成的多边线不止一条,因此需要对生成的多边线进行拆分处理。拆分处理是基于多边线上的数据有足够的密度,因此可通过计算多边线各段的长度(如图 7 所示长度的正态分布)来进行拆分。实验结果证明,这种方法对一般的点云数据是可行的。图 8 是边界特征点生成多边线的示例。

拆分的具体过程如下:

首先计算多边线各段的长度得到一个长度序列 $\boldsymbol{M}=(l_1,l_2,\cdots,l_n)$,计算 $\boldsymbol{M}$ 平均长度 $\bar{l}$,再得到一个差序列 $\boldsymbol{M}'=(l_1',l_2',\cdots,l_n')$,其中,$l_i'=|l_i-\bar{l}|$。计算差序列 $\boldsymbol{M}'$的平均值$\bar{l'}$,若 $l_i'>\bar{l'}+\sigma$,则认为该多边线在此线段上可以拆分。

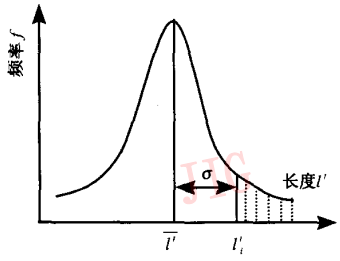


图 7 长度差的正态分布图
(灰色地带为多边线之间的线段)

Fig. 7 Normal school of distance's difference
(Gray zone is abnormal difference)

(a) 边界点 (b) 原边界多边线 (c) 拆分后的边界多边线

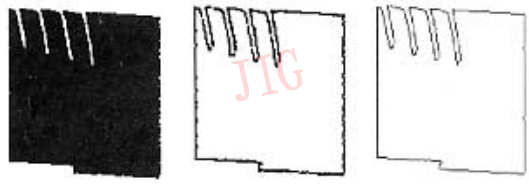
图 8 边界点生成多边线

Fig. 8 Polyline from points

6 实验结果

应用本文检测边界特征点的方法进行实验,实验结果如图 9 ~ 图 12 所示。图 9 为点云数据边界的提取和排序完后的多边线边界;图 10 为带有 2 个边界轮廓的点云数据边界提取;图 11 为同时具有内边界和外边界的点云数据边界提取;图 12 为利用本文算法实现的保留边界数据点的点云数据压缩与没有保留边界数据点的压缩算法的比较。从图中可以明显看出,本文提出的边界数据检测算法可以有效地保持原有边界数据点的完整性,这在实际应用中具有重要意义。

经过对不同数据实验结果的分析,*K*-Nearest Points 和标准差阈值的选择对计算结果影响较大。*K*-Nearest Points 的点数取值应该取决于点云数据的密度和均匀性,当点云数据密度大时该值可取小些;当点云数据比较稀疏时该值可取大些,但都应该能满足正确构造最小二乘平面。经验表明一般该值取 10 ~ 30 之间是一个较好的选择。标准差阈值则取决于点云数据边界的空间复杂程度,若点云数据边界很曲折且密度又大时该值可以取小些;当点云数据的边界比较平直时,该值可以取得大些。



(a) 原始数据点云 (b) 边界提取 (c) 边界的多边线形式

图 9 点云数据边界提取及边界的多边线形式

Fig. 9 Boundary extraction of point cloud and polyline creation

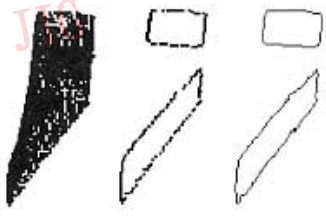
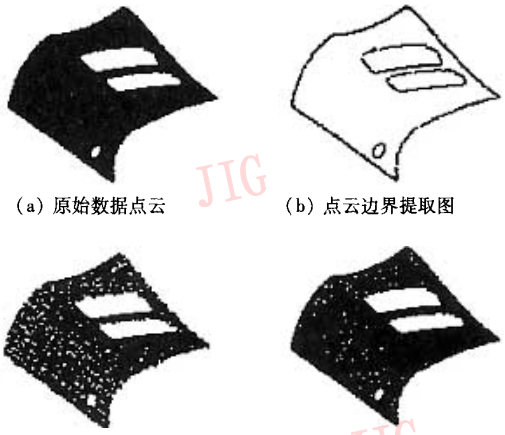


图 10 带有两个外边界的点云数据边界提取

Fig. 10 Boundary extraction of point cloud with outer loop boundary

图 11 同时具有内边界和外边界的点云数据边界提取

Fig. 11 Boundary extraction of point cloud with inner and outer loop boundary



(a) 原始数据点云 (b) 点云边界提取图

(c) 无边界保留算法的数据压缩 (d) 有边界保留算法的数据压缩

图 12 不同边界数据处理的点云数据压缩算法比较

Fig. 12 Comparison of point cloud compression due to boundary preservation algorithm

7 结　论

针对点云数据压缩过程中边界数据丢失问题,提出了一种点云边界特征点检测算法,该算法利用BSP 树对数据点进行处理,并由数据点邻域构成的最小二乘平面计算投影点的角度标准差阈值,从而确定点云数据的边界特征点,对边界特征点进行排序,为后面的 3 维模型的反求造型做铺垫。该算法的主要贡献在于在进行空间散乱点云数据压缩时能保留足够多的边界数据点,从而确保大比例压缩过程中物体外形拓扑关系的正确性,避免对原始数据造成的失真。同时本文提出的边界点检测算法在点云数据特征提取、空洞区域的数据填充等方面也具有重要意义。在此基础上,下一步的工作将主要集中在边界曲线的自动生成以及内部特征数据点的自动检测算法方面。

参考文献(Reference)

1 Varady T, Martin R R, Cox J. Reverse engineering of geometric models-an introduction[J]. Computer-Aided Design, 1997,**29**(4): 255 ~ 268.

2 Cheng Z Y. B-Spline surface reconstruction theory based on triangular surface model and it's application in reverse engineering[D]. Hangzhou: Zhejiang University, 2001. [陈志杨. 基于三角曲面原型的 B 样条曲面重构理论及其在反求工程中的应用研究[D]. 杭州:浙江大学,2001.]

3 Cheng Z Y. Reverse engineering CAD modeling based on triangular surface[J]. China Mechanical Engineering, 2003,**14**(8):698 ~ 700. [陈志杨. 基于三角曲面的逆向工程 CAD 建模方法[J]. 中国机械工程, 2003,**14**(8): 698 ~ 700.]

4 Pauly M, Gross M. Spectral processing of point-sampled geometry [A]. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques[C], New York: ACM Press, 2001:379 ~ 386.

5 Lee In-Kwon. Curve reconstruction from unorganized points[J]. Computer Aided Geometric Design, 2000,**17**(2): 161 ~ 177.

6 Zhou Y L, Xiong P R, Zhu H. An improved algorithm about the closest pair of points on plane set[J]. Computer research & development, 1988,**35**(10):956 ~ 960. [周玉林,熊朋荣,朱洪. 求平面点集最近点对的一个改进算法[J]. 计算机研究和发展, 1998,**35**(10): 956 ~ 960.]