预定义形状的 3 维表面切割算法研究

梁荣华[1] Clapworthy Gordon[2] 潘志庚[3] 吴福理[1]

[1](浙江工业大学信息工程学院,杭州 310032) [2](Department of Computing & Information Systems, University of Luton, UK)

[3](浙江大学 CAD&CG 国家重点实验室,杭州 310027)

摘 要 切开外部物体的表面去显示里面的物体,在可视化研究领域具有广泛的应用前景。以前采用计算机图形学的方法是用透明显示,或者要求切割的形状是规则的。本文重点研究了任意形状的切割,首先用户在 2 维的空间中定义一个形状,然后将该形状投影到外面 3 维物体上再做切割,该算法还解决了连界上的一些锯齿形问题,实验结果表明该算法是有效的。

关键词 切割 锯齿问题 OBB 树 投影

中图法分类号:TP391 **文献标识码**:A **文章编号**:1006-8961(2006)12-1865-05

Predefined Shape Cutaway on 3D Surface

LIANG Rong-hua[1], Clapworthy Gordon[2], PAN Zhi-geng[3], WU Fu-li[1]

[1](*College of Information Engineering, Zhejiang University of Technology, Hangzhou* 310032)

[2](*Department of Computing & Information Systems, University of Luton, UK*)

[3](*State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou* 310027)

Abstract Cutting away part of a 3D outer surface to show some interior objects hidden beneath it can play an important role in visualization and virtual reality. Previous methods have mainly aimed at showing the interior objects, and the shape of the hole has been considerably less important. In this paper, we propose a novel approach to cutting a specific hole in a surface. The user first defines the required shape on the surface; the algorithm then superimposes the shape on the surface and implements the cutaway. Jittering problems associated with the boundary generated by the above steps are resolved and experimental results demonstrate the efficiency of the algorithm.

Keywords cutaway, jittering problems, OBB-tree, projection

1 引 言

切割物体表面可以让用户观察到移除了外部表层组织的不透明物体的内部结构。一个著名的技术就是计算机图形学中 α 透明处理[1]。然而,透明处理只是一种擦除。也有些工作采用透视技术而非切割方法,在非真实感的图像渲染中来显示内部空间的组织关系[2~4]。其他的工作着眼于模拟不同的渲染方法,如色调阴影法和轮廓渲染法[5],还有一些工作是采用硬件进行渲染的方法[6]。然而,在一些应用中,切割的应用是很普通的,如在外科手术中,需要模拟皮肤的切割形状去显示内部的肌肉。

Diestraten 等人[7]提出通过定义若干规则来进行切割的方法。这些规则适用于基于计算机图形学的绘制,但不局限于一种特定的渲染类型,而是能与许多著名的绘制技术完美结合。然而,Diestraten 使用的剖面形状是规则的多边形。我们的应用是关于整形外科手术的术前计划,因此希望在皮肤上产生一个切口然后进行手术,首先,当皮肤从切口回缩的时候能反映出外科医生所预期的形状;其次,利用一些参数来控制这些形状,它能反映由于不同病人的

基金项目:浙江省自然科学基金项目(Y105095,Z603262)

收稿日期:2006-05-30;改回日期:2006-09-05

第一作者简介:梁荣华(1974~),男,副教授。2003 年于浙江大学获计算机科学与技术专业博士学位。主要的研究方向为医学图像可视化、计算机图形学、计算机视觉。E-mail:rhliang@zjut.edu.cn

身体而产生的皮肤拉伸特性。显然,在这种情况下,使用多边形的切割来描述一个皮肤切口是不适当的。

本文提出了一个新的实时切割算法,不失一般性,该切割的外部表面是由 3 维多边形网孔构成。这个形状首先在 2 维平面被定义,然后对物体表面进行 OBB(oriented bounding box)树[8]分析,把用户自定义的形状投影到 3 维空间,最后进行切割。上述步骤也有效解决了切口的锯齿性边界问题。

2 算法描述

2.1 假设

不失一般性,首先切口只作用于外部表面,内部组织没有被切开,外部物体由 3 维多边形网孔组成,假定其为三角网格,并且没有自交叉。表面的切口形状可以是曲线、多边形等,但是必须是由点与线构成,并且不自相交。综合起来,假设如下:

(1) 内部和外部物体必须明确区分开;

(2) 外部物体由 3 维三角网格组成,采用 *OS* 表示;

(3) 外部物质不存在自相交;

(4) 切开表面的形状在同一平面上,且由线与点构成,用 *US* 表示;

(5) 该形状 *US* 不存在自相交情况。

2.2 算法概述

算法由 4 步组成,如图 1 所示。尽管也有其他的树形包围盒(例如 AAAB 树,球体树)来逼近 3 维物体,但是 OBB 树可更为紧密围住目标(见图 2)。此外,根据需要可以在 OBB 树的节点上添加有用的数据。为了快速在外表面上产生 *US* 的近似形状,采用了 BSP 树(空间二分树)来加速搜索 *US* 与 *OS* 中最接近的点(也就是 *US* 在 *OS* 外表面的投影点)然后用线剪取算法(clipping algorithm)形成一个切割(cutaway),与 Diepstraten 等人提出的切割非常相似。为了修正边界的锯齿现象,需要相应地将相邻的跨两个三角形网格的点细分为更多的投影点。

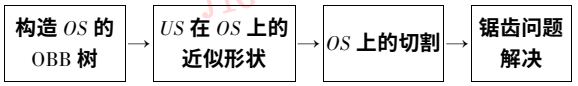
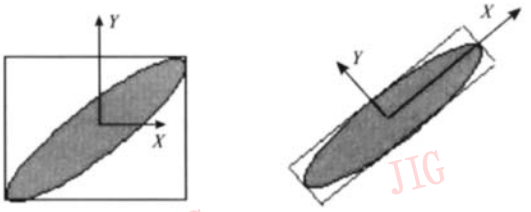


图 1 基本算法流程

Fig. 1 Algorithm outline



(a) AAAB (b) OBB

图 2 AAAB 与 OBB 的比较

Fig. 2 Comparison of AAAB and OBB

2.3 外表面的 OBB 树分析

OBB 要求满足:

(1) *OS* 的第 i 三角形的质心为

$$m_i = (p_i + q_i + r_i)/3$$

其中,p_i,q_i,r_i 是三角形的顶点;

(2) μ 是整个 *OS* 的质心

$$\mu = \frac{1}{3n}\sum_{i=1}^{n}(p_i + q_i + r_i) \tag{1}$$

其中,n 是 *OS* 所有的三角形个数;

(3) 3×3 阶协方差矩阵 C 如下:

$$C_{j,k} = \frac{1}{3n}\sum_{i=1}^{n}[\bar{p}_i^j\bar{p}_i^k + \bar{q}_i^j\bar{q}_i^k + \bar{r}_i^j\bar{r}_i^k] \tag{2}$$

其中,$\bar{p}_i = p_i - \mu$,$\bar{q}_i = q_i - \mu$,$\bar{r}_i = r_i - \mu$。

OBB 应用了一种统计方法——主成分分析,用以发现最佳的 3 个主轴(主成分因子),使从质心到每个顶点的距离(方差)最小。因为 C 是对称矩阵,所以它的特征向量相互正交。C 的 3 个特征向量中的两个就是具有最大方差和最小方差的轴,标准化的特征向量用作基础轴。找到每根基础轴上的极值点,在基本向量的方向上,将包容盒按尺寸大小排序,进而包含那些极值点。而 OBB 树是这样一种包容盒子的树形结构:OBB 在更深的水平上限制更小的空间区域。已经证明,在刚性动作的复杂模型中冲突检测非常有效,它能更好地逼近 3 维物体。

为了构建 OBB 树,采用递归的自顶向下的过程。首先,用上面的方法构建 OBB 树的根。然后创建两个子 OBB 树,用分割面把数字网络大致一分为二。这个过程将不断继续,直至迭代收敛结束,或者因找不到分割面而结束。

为了实现上述过程,找到原先设定在每个轴上的三角形的最大和最小范围,然后用一个与 OBB 其中一轴正交的平面分割 OBB 的最长一个轴,根据多边形中心点在这个平面的那面来分割多边形,选择轴上的细分坐标点作为顶点的均值点。通常,OBB

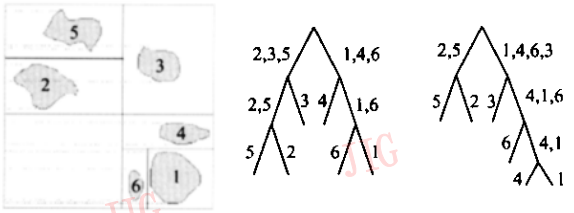
树用于碰撞检测。将其进行改进，记录（保留）第 i 个节点上的每个质心 m_i，通过组织数据，找到最接近的点，以形成后面的 US。因此，OS 的 OBB 树可以表述为

$$OS_{\mathrm{OBB}} = \{n_1, n_2, \cdots, n_m\} \tag{3}$$

其中，节点 $n_i = \{N_{\mathrm{OBB}_i}, m_i\}$，$N_{\mathrm{OBB}_i}$ 是 OBB 树的标准节点，m 是 OBB 树的节点个数。

2.4　*US* 在外表面的近似形状计算

当计算 OBB 树描述的 OS 近似形状时，需要考虑能够快速获得 OBB 树上所有节点的质心的几何（模型）数据库的技术。图 3 描述了 2 维空间中的“空间二分”的概念。



（a）原始空间位置　（b）Y-X 轴分割排序　（c）X-Y 轴分割排序

图 3　采用 BSP 树划分

Fig. 3　Divide space with BSP

采用空间二分法（BSP），可以把空间分成更小的区域。由于每个区域总与一个物体或者一组物体有关，当从上自下沿着二分树的分支搜索时，可以避免不必要的检查。采用物体的最大和最小坐标来实现空间分割。举例说，如图 3（b），物体的最大和最小坐标投影到 Y 轴上，接着在 Y 轴上对物体进行排序和分割，然后在 X 轴上对物体排序。

根据 OS_{OBB} 上节点的所有质心 m_i，在 X、Y、Z 轴上分别构建 BSP。BSP 树可以表现在 3 个不同的方向上，就是 X-Y-Z，Y-X-Z 和 Z-X-Y

$$OS_{\mathrm{BSP}} = \{T_{\mathrm{BSP}_1}, T_{\mathrm{BSP}_2}, T_{\mathrm{BSP}_3}\} \tag{4}$$

2.1 节假设 4 指出，US 由点（在轮廓曲线上）与直线段组成。采用 BSP 树，可以加速 US 到 OS 表面上的最近点的匹配。

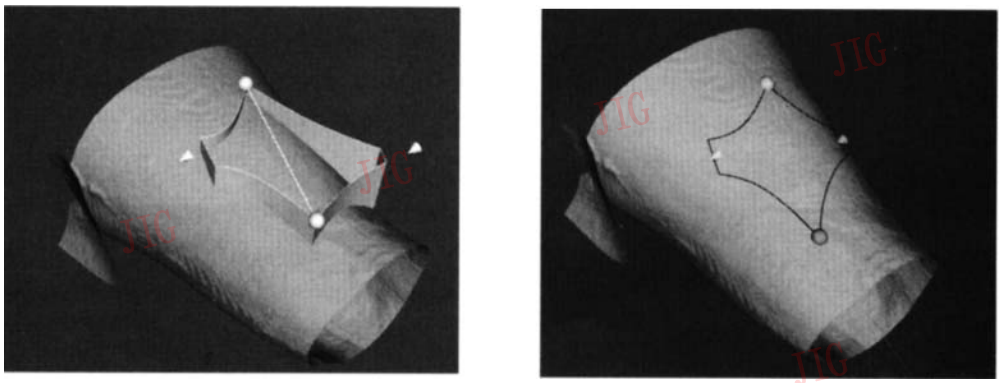
可以找到在 T_{BSP_k}（$k = 1, 2, 3$）上指向 US 上的 P 最接近的节点 i，如：节点 $T_{\mathrm{BSP}_k^i}$。如果 $T_{\mathrm{BSP}_1^i}$，$T_{\mathrm{BSP}_2^j}$，$T_{\mathrm{BSP}_3^k}$ 是 P 最接近的 BSP 节点，i，j，k 不必相同，从 P 到 $T_{\mathrm{BSP}_1^i}$ 的最小距离（方差）是 OS 上 P 的近似投影点，则 OS 表面上的 P 的近似投影点 P' 可以表述为

$$\underset{k}{\mathrm{Arg}}\ \min_{k=1}^{m}\left(\sum_{i=1}^{3}(w_i T_{\mathrm{BSP}_k^i}) - \sum_{i=1}^{3}(w_i P_i)\right) \tag{5}$$

其中，w_i 是每个 BSP 树节点的权。

投影结果如图 4 所示。

投影后，切割算法的关键点就是在 US 投影区域内裁剪。用 US' 表示投影在 OS 上的 US 形状，US' 就作为裁剪窗口，即在 US' 内的区块将被剪掉。该算法已经包含在可视化软件开发包 VTK（Visualisation toolkit）中[9]，能够自动完成裁剪。



（a）投影的过程　（b）投影结果（用黑线表示）

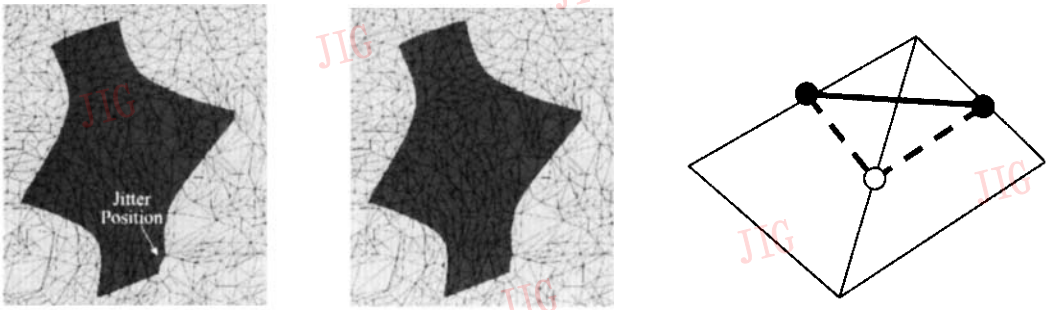
图 4　*US* 在 *OS* 上的投影

Fig. 4　Projection of *US* on *OS*

2.5　锯齿问题的解决

如果在两个毗连的投影点之间，跨越了两个三角形，那么边界的锯齿问题就可能发生。

在图 5（a）、（b）中，黑色区域表示切割的形状，其中，（a）显示存在的基准误差，（b）表示基准误差消除后的边界。让 US 在 OS 上投影的两个点之间生成一些附加“投影”点，用以消除在那些边界上的基准误差。目前，用来消除锯齿误差的附加投影点



(a) 锯齿问题 (b) 去掉锯齿之后 (c) 锯齿问题的解决示意图

图 5 边界锯齿问题

Fig. 5 Jittering problem on the bound

是通过将相邻两个点再细分成更多点的方式来计算的。图 5(c)中的黑点表示投影的点,白点表示新产生的点。在算法里,如果在两个相邻投影点之间的长度是超过一个极限,他们之间就会产生新的点。

3 实验结果

算法已经在 C ++ 环境下用 OpenGL 和 VTK 实现[9],并应用到全髋骨置换手术(THR)的手术前计划项目[10],在该项目中,提供了一种真实感的手术环境:包括触觉、动作、立体视觉和语音控制,外科医生可以据此做手术计划,并测试任何计划实施的手术动作对病人在术中和术后可能造成的伤害,其中割开皮肤是手术的第一步。根据病人皮肤伸展时的可能特性,对切开后的皮肤形状进行虚拟。预定义的形状包含一些参数,这些参数能够调整切口的形状。

图 6 展示了在 THR 应用中的具体形状,它由两条直线和 4 条可调整的样条曲线组成。图 7 将本文算法与 Diepstraten 等人的算法[7]进行了比较,Diepstraten 等用的切割仅由 6 条直线组成,显然,他们的方法不适于 THR 的手术前计划项目。

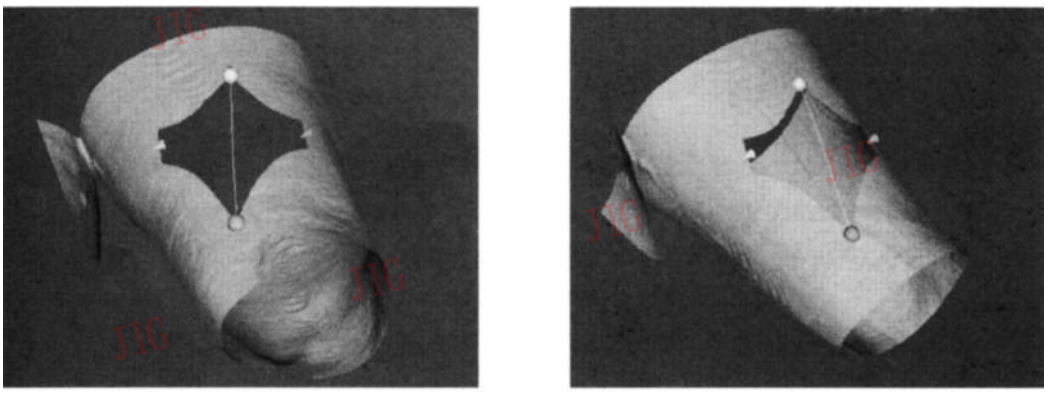
表 1 列出了测试中计算所用的时间。本文算法需要更多的计算时间,这主要是因为计算形状的投影所造成的。

表 1 计算速度比较

Tab. 1 Comparison of computation speed

三角形数	顶点数	所耗时间(ms)	
		Diepstraten 等人算法	本文算法
4 532	8 521	0.12	0.21
23 800	35 263	0.67	1.10
3 000	6 421	0.07	0.15

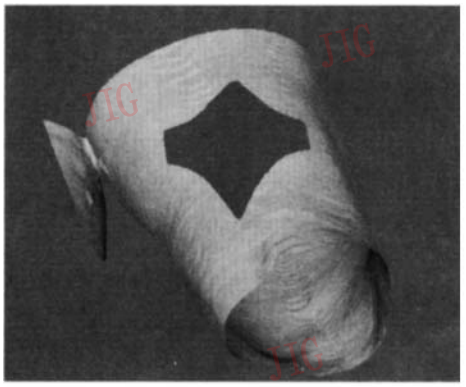
为了计算 OBB 树,计算的复杂度是 $O(n\log^2 n)$,其中,n 是输入三角网格的数量。如果 m 表示 US 上点的数量,在表面上形成形状的计算复杂度就是 $O(m\log n)$。所以本文算法在理论上是实时的。



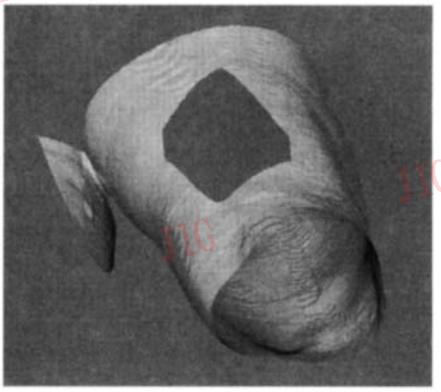
(a) 切割的结果 (b) 对(a)另一个视图的结果

图 6 实验结果

Fig. 6 Experimental results



(a) 本文算法结果



(b) Diepstraten 算法结果

图 7　算法比较

Fig. 7　Comparison of algorithms

4　结　论

本文提出了一种全新的算法，来完成物体的外表面的预定义形状的实时切割，算法的优势在于表面切口的形状比以前算法中的规则形和多边形更加通用。

我们采用并且扩展 OBB 树来概算外表面和用 BSP 树形成在表面上的预定义形状。解决了与边界相关的锯齿问题。并在普通 PC 机上实验了这种算法实时工作的情况。这种算法已经应用到 THR 的手术前计划。

参考文献(References)

1　Blythe D. SIGGRAPH 1999 Course Notes：Advanced Graphics Programming Techniques Using OpenGL[CD]. New York：Addison-Wesley，1999.

2　Dooley D，Cohen M F. Automatic illustration of 3D geometric models：surfaces[J]. IEEE Computer Graphics and Applications，1990，**13**(2)：307～314.

3　Diepstraten J，Weiskopf D，Ertl，T. Transparency in technical illustrations[J]. Computer Graphics Forum，2002，**21**(3)：317～325.

4　Hamel J，Schlechtweg S，Strothotte T. An approach to visualizing transparency in computer-generated line drawings [A]. In：Proceedings of IEEE Information Visualization 1998[C]，London，UK，1998：151～156.

5　Gooch A，Gooch B，Shirley P，*et al*. A nonphotorealistic lighting model for automatic technical illustration[A]. In：Proceedings of SIGGRAPH[C]，Orlando，USA，1998：101～108.

6　Praun E，Hoppe H，Webb M，*et al*. Real-time hatching[A]. In：Proceedings of SIGGRAPH[C]，Los Angeles，USA，2001：579～584.

7　Diepstraten J，Weiskopf D，Ertl T. Interactive cutaway illustrations[J]. Computer Graphics Forum，2003，**22**(3)：523～532.

8　Gottschalk S，Lin M，Manocha D. OBB-Tree：A hierarchical structure for rapid interference detection[A]. In：Proceedings of SIGGRAPH[C]，New Orleans，USA，1996：171～180.

9　The visualization Toolkit [EB/OL]. http://public.kitware.com/VTK/index.php，2002.

10　European Projects [EB/OL]. http://www.beds.ac.uk/root/research/irac/ccgv/euprojects/index，2004.