三维模型轮廓线抽取算法

吴亚东 刘玉树
(北京理工大学计算机科学工程系,北京 100081)

摘 要 虽然三维模型的轮廓线在图形交流中起着重要的作用,但由于轮廓线是视点依赖的,当物体在空间运动时,用现有算法从复杂的三维模型中抽取轮廓线需耗费大量时间,为了提高抽取轮廓线的速度,提出了两种新的抽取三维模型轮廓线算法.这两种算法是先利用轮廓线的局部极值特性来获得部分轮廓边,然后利用轮廓线的连通性,通过简单的比较运算,即可获得三维模型的外部轮廓线.实验结果表明,该两种算法都可快速获得三维模型的外围轮廓线,最后还将本算法与相关算法进行了详细的比较分析.

关键字 三维模型 轮廓线 抽取 连通性 绘制
中图法分类号:TP391 **文献标识码**:A **文章编号**:1006-8961(2001)02-0191-04

Extracting Silhouettes from 3D Models

WU Ya-dong, LIU Yu-shu
(*Department of Computer Science and Engineering, Beijing Institute of Technology, Beijing* 100081)

Abstract Silhouettes play a very important role in graphics communication. Because silhouettes are view-dependent, it is expensive to extract silhouettes for previous techniques when the object is moving in space. This paper presents two algorithms of extracting silhouettes from 3D models. By exploiting the property of local maximum value and connectivity of silhouettes, the methods first identify part of silhouettes, then extract external silhouettes of 3D models by simple comparisons. The first method divides the model into some zones evenly in X and Y direction. Then it calculates the maximum edges of the zones in X and Y direction. These edges are silhouette edges. By searching the neighbor maximum edges of the silhouette edges, it finds the silhouette of the model. The second approach begins from the top edge in Y direction. By comparing the angles between the silhouette edge and its neighbor edges, it obtains the silhouette easily. Experimental results illustrate the efficiency of the methods. The algorithms and previous methods are compared and analyzed at the end of the paper.

Keywords 3D models, Silhouettes, Extraction, Connectivity, Rendering

0 引 言

在交互式三维绘制中,模型的复杂度与绘制速度始终存在着矛盾,但为获得较好的显示效果,就必须提高模型的复杂度,因而这就不可避免地会降低模型的绘制速度.为了缓解这一矛盾,相继开发了 LOD 技术和基于图象的绘制技术.三维模型的特征值抽取不仅是近年来一种相当引人注目的技术[14],而且它对 LOD 技术和基于图象的绘制技术也具有重要意义.由于轮廓线是一种特征值,其形式简洁,而内涵丰富,因此根据轮廓线就可以识别出对象的三维结构[35].另外,由于轮廓线在图形交流中所起的巨大作用,因此这一技术在科学可视化、三维场景的快速绘制以及地理信息系统中有着广阔的应用前景.

1 三维模型的轮廓线

为讨论方便起见,这里首先给出轮廓点的定义.

定义 由多边形组成的模型 M,其表面 F 上一点 P,设 P 点的法向量为 $\boldsymbol{n}$,$\boldsymbol{E}$ 为视线,若 $\boldsymbol{E}\cdot\boldsymbol{n}=0$,即 $\boldsymbol{E}$ 与 $\boldsymbol{n}$ 的夹角为 90°,则 P 为轮廓点.

一般轮廓点均位于组成模型的多边形的边上.

基金项目:高等学校博士学科点专项科研基金资助课题
收稿日期:2000-01-19 改回日期:2000-06-06

当轮廓点位于多边形的某条边上时,则称该边为轮廓边.轮廓点投影到投影平面上,则形成轮廓线.这种轮廓线分为外部和内部两种,其中,外部轮廓线组成了物体投影与背景的边界,且为一条二维连通曲线;而位于外部轮廓线内部的轮廓线为内部轮廓线.

由于轮廓线的边数比三维模型的总边数要少得多,所以用轮廓线取代三维模型会大大减少绘制时间,然而,从三维模型获得轮廓线的思路虽然简单,但由于要进行大量的浮点乘法运算,其计算量是相当大的.如 Zhang 给出的公式[6],当且仅当

$$(\boldsymbol{n}_1 \cdot (v - e)(\boldsymbol{n}_2 \cdot (v - e))) \leq 0 \qquad (1)$$

则该条线即为轮廓边,其中,v 为该边上一点;e 为视点;$\boldsymbol{n}_i (i = 1,2)$ 为共享该边的两个面的法向量.由于轮廓线是视点依赖的,因此当物体在空间移动时,必须连续进行判断.对于复杂度较高的模型,特别是在透视投影的情况下,要实时对每条边进行这样的计算,其计算量是很难接受的.

Markosian 提出一个实时性较好的算法[7],该算法通过离散检查较少数量的边,得到部分轮廓边后,利用连通性找出所有的轮廓线.为增大离散检查时获得轮廓边的概率,首先将所有边按共享该边的两个三角形之间的夹角大小进行排序.这一算法尽管减少了需检查的边数,但其判断轮廓边的方法仍需进行耗时的浮点乘运算,并且,该算法并不能保证很快搜索到部分轮廓边.

为减少运算时间,本文给出两种从三维模型抽取外部轮廓线的算法(如无特别说明,本文中所指的轮廓线均为外部轮廓线),这两种算法均是先利用轮廓线的局部极值特性来获得部分轮廓边,然后利用连通性找出轮廓线.在此过程中,无须进行复杂的浮点乘运算,从而有效地减少了轮廓线的抽取时间.

2 算法描述

由于模型的轮廓边为一条连接一个可见面与一个不可见面的连接边,因此从该模型在投影面上的投影来看,这种轮廓边无疑为该边所在局部区域的极值边(极上、极下、极左或极右边),因而只要找到轮廓线中的一部分边,再根据轮廓线的连通性,就可以获得所有相邻的极值边,即可以获得三维模型的轮廓线.按照这一思想,本文提出了两种在部分轮廓边的获得和相邻边的检查方法上都有所不同的算法.

2.1 算法 1

首先将模型上下、左右分为若干个区间,通过简单的比较,即可获得每一区间的极值边,即轮廓边,然后根据轮廓线的连通性,找出所有相邻的极值边,即获得三维模型的轮廓线.

该算法分为 3 个步骤:①在模型变换时求出模型在投影平面上的极值(极上、极下、极左和极右);②划分区间,求出每一区间的两条轮廓边;③根据连通性得到三维模型的轮廓线.

2.1.1 部分轮廓边

按照 OpenGL 的坐标系,设视点位于 Z 轴(其他坐标系可类似处理).若设视点不动,则在旋转、平移模型的同时,即可求得 X 方向和 Y 方向的极大和极小值.将模型在 X 方向和 Y 方向均匀划分为若干个区间(如图 1 所示),区间索引为 $[0,\cdots,m]$;依次检查模型的每条边,因为已获得 X 方向和 Y 方向的极大值和极小值,故可通过索引来求得该边所在的区间.设 X 方向极小值及极大值分别为 $x_{\min}$、$x_{\max}$,X 方向区间数为 N,则区间大小 l 为

$$l = \frac{x_{\max} - x_{\min}}{N} \qquad (2)$$

设边 e 的两个顶点为 v_1、v_2,顶点的 x 值为 x_{v_1}、x_{v_2},则边 e 在 X 方向的区间索引为

$$\frac{\min(x_{v_1}, x_{v_2}) - x_{\min}}{l}$$

而且模型的各条边在 Y 方向的区间索引也可类似求得.

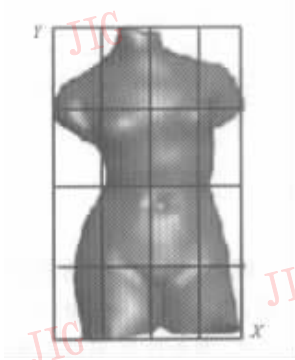

图 1 将模型在 X 方向和 Y 方向划分

经过对模型所有边的一次扫描,即可得到 X 和 Y 方向各区间的极大边和极小边(在 OpenGL 坐标系中,某一区间中 Y 方向极大边和极小边即指该区间中最上和最下的边,且 X 方向也与 Y 方向类似,如图 2 中的 AB 边即为某一区间中 Y 方向的极大边),这些极值边即为各区间的部分轮廓边.

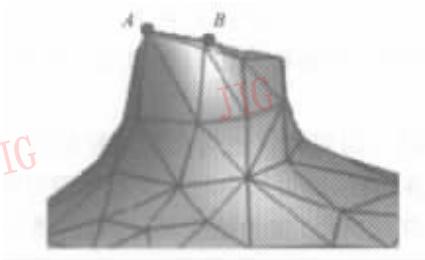

图 2　Y 方向最大边 AB

在实验中还发现,在 X 和 Y 方向的区间数只需不足模型总边数的 1%,就可以获得比较完整的轮廓线,这一点与文献[7]论述类似.

2.1.2　获得轮廓线

在得到 X 和 Y 方向每一区间的两条轮廓边后,即可根据轮廓线的连通性,依次检查 X 和 Y 方向的每一区间的两条极值边,其中,对极大边,则搜寻与该边相邻的极大边;对极小边,则搜寻与该边相邻的极小边;对获得的新轮廓边则接着进行相邻边的检查,直至到达下一区间为止. 这样,在对所有区间进行一次检查后,即可获得该模型的外围轮廓线.

2.2　算法 2

如前所述,从模型在投影面上的投影来看,一条轮廓边为该边所在局部区域的极值边,那么,只要找到一条这样的轮廓边(如当前投影平面上位置最高的一条轮廓边),则根据轮廓线的连通性,就可获得所有相邻的极值边,即获得三维模型的轮廓线.

与算法 1 相同,算法 2 仍然按照 OpenGL 的坐标系,设视点位于 Z 轴. 若设视点不动,则在旋转、平移模型的同时,即可求得 Y 方向位置最高的一条边,这条边即为相对于该视点的一条轮廓边. 在获得最高位置的一条轮廓边以后,就可以按顺时针或逆时针方向找到相邻的下一条轮廓边(如图 3 所示),如在图 3 单位圆中,已知 a 为一条轮廓边,即可按顺时针方向查找下一条与边 a 在顶点 v 相邻的轮廓边. 由于从边 a 按逆时针方向到边 b 的夹角最大,显然边 b 即为下一条与边 a 相邻的轮廓边. 如此循环直至到达与起始时 Y 方向位置最高的轮廓边时,则停止搜索,此时得到的所有边即为相对于该视点三维模型的轮廓边.

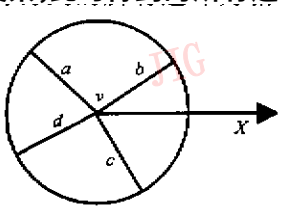

图 3　与轮廓边 a 相邻的边

在查找与边 a 相邻的轮廓边时,应首先求得边 a 及其所有相邻边与 X 轴的夹角,如设边 a 与 X 轴的夹角为 α,相邻边 b 与 X 轴的夹角为 β,如按逆时针方向,则边 b 与边 a 夹角 θ 为

$$\begin{cases} \theta = \beta - \alpha & \text{if}(\beta - \alpha > 0) \\ \theta = 2 \times \pi + \beta - \alpha & \text{else} \end{cases}$$

算法程序流程如下:

```
void FindContour(Model * model,Contour * contour)
 //查找模型的外部轮廓线
 {
   edge_start = FindMaxEdge(model);
     //获得一条极值边,即轮廓边
   AddToContour(model,contour, edge_start);
     //将该边加入到轮廓线列表中
   point_start = EdgePointStart(model, edge_start);
     //从 point_start 开始检查相邻边
   edge_next = FindNextEdge(model, edge_start, point_start);
     //获得一条相邻的轮廓边
     while(edge_next ! = edge_start)
      {
        AddToContour(model, contour, edge_next);
        point_start = EdgePointStart(model, edge_next);
        edge_next = FindNextEdge(model, edge_next, point_start);
      }
 }
```

3　实验结果与结论

以上算法已用 C 语言实现,部分实验结果见图 4、图 5 和图 6. 图 4(b)由算法 1 生成,图 5(b)和图 6(b)由算法 2 生成. 图 4(a)、图 5(a)、图 6(a)的三角形个数分别为 1 418、69 451 和 5 804.

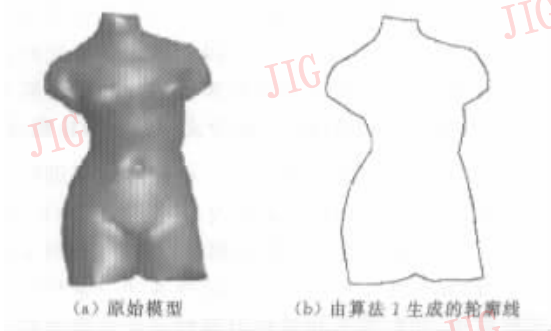

图 4　维纳斯模型轮廓线抽取结果

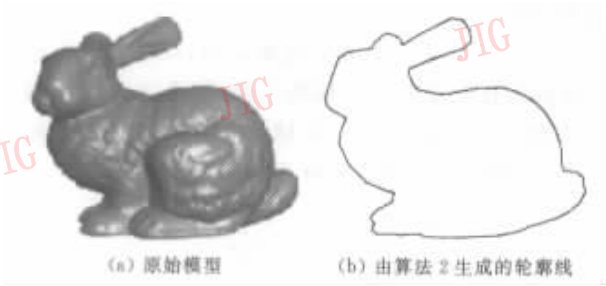

图 5 兔子模型轮廓线抽取结果

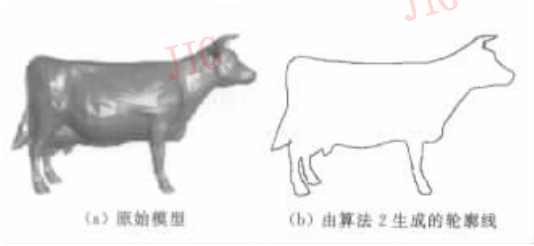

图 6 牛模型轮廓线抽取结果

三维模型的轮廓边在投影平面上可能出现交叉,即可能一条轮廓边的所有相邻边都不是轮廓边,这样将致使轮廓边提取十分困难,而算法 1 在 X 和 Y 方向有多个区间正好对应处理了轮廓边的这一特性.通过实验还发现,算法 2 在这一方面也处理得很好,即当一条轮廓边的所有相邻边都不是轮廓边时,算法 2 会在搜索了几条非轮廓边以后,即能正确查找到下一条轮廓边,图 5(b)和图 6(b)就说明了算法 2 的有效性.相比而言,由于算法 1 需要指定 X 方向和 Y 方向的区间个数,因此运算较为复杂,由此可见,算法 2 比算法 1 更易于操作.

设三维模型有 n 条边,相对于某一视点的轮廓线有 m 条边,$m = O(\sqrt{n})$,文献[6]的方法需对每一条边按公式(1)计算,共需 $7n + 2/3 \times 6n = 11n$ 次乘法运算,而算法 1 则只需 $2n$ 次除法运算和 $O(m)$ 次比较运算,显然算法 1 大大减少了计算量.与文献[7]相比,由于算法 1 不需要预先对所有边按共享该边的两个面的夹角大小排序,因此可以更有效地搜索到部分轮廓线,并且判断轮廓边的方法也要简单得多.用文献[7]中的算法,要搜索一条起始轮廓边,需检查 $O(\sqrt{n})$ 条边,而算法 2 则在对模型进行变换的同时,就搜索到起始轮廓边,其复杂度为 $O(1)$.实际上,除非对所有的 n 条边都进行判断,否则文献[7]中的算法并不能保证一定能搜索到一条起始外部轮廓边,即它并不知道所得到的起始轮廓边是外部,还是内部的,而算法 2 则可以保证起始轮廓边即为外部轮廓边.当搜索到起始轮廓边后,算法 2 与文献[7]算法搜索相邻轮廓边的时间复杂度相同,均为 $O(m)$.

本文给出的两种算法均是利用轮廓线的局部极值特性和连通性,通过简单的比较运算即可获得三维模型的外部轮廓线.实验结果显示,两种算法都可快速获得三维模型的外围轮廓线,并达到了预期目的.

参 考 文 献

1 Luebke D, Erikson C. View-dependent simplification of arbitrary polygonal environments. In: Whitted T. ed. Proceedings of SIGGRAPH '97, Computer Graphics Proceedings, Annual Conference Series. Los Angeles: Addison Wesley, 1997: 199208.

2 Gu X, Gortler S, Hoppe H *et al*. Silhouette mapping. Technical Report TR-1-99, Department of Computer Science, Harvard University, March 1999.

3 Gooch A, Gooch B. Using non-photorealistic rendering to communicate shape. In: SIGGRAPH '99 Course Notes, Section 8, Los Angeles, 1999.

4 Ma K, Interrante V. Extracting feature lines from 3-D unstructured grids. In: Yagel R, Hagen H. ed. IEEE Visualization '97. Phoenix AZ: IEEE, 1997: 285292.

5 Christensen P H. Contour rendering. Computer Graphics, 1999, 33(1): 5860.

6 Zhang H, Hoff K. Fast backface culling using normal masks. In: Cohen M, Zeltzer D ed. 1997 Symposium on Interactive 3-D Graphics. Providence Rhode Island: ACM SIGGRAPH, 1997: 103106.

7 Markosian L, Kowalski M, Trychin S et al. Real-time nonphotorealistic rendering. In: Whitted T. ed. Proceedings of SIGGRAPH '97, Computer Graphics Proceedings, Annual Conference Series. Los Angeles: Addison Wesley, 1997: 415420.

吴亚东 1974 年生,北京理工大学计算机科学工程系博士生.主要研究方向为交互式绘制、模型简化、非真实感图形绘制、虚拟现实.

刘玉树 1941 年生,北京理工大学计算机科学工程系教授,博士生导师.主要研究方向为人工智能、图象处理、计算机图形学、地理信息系统.