有限元网格体绘制中的剖切算法

杨晓松 顾元宪 李云鹏 关振群
(工业装备结构分析国家重点实验室;大连理工大学工程力学系, 大连 116024)

摘 要 为了解决体绘制中的遮挡问题和加快复杂剖切体的剖切操作,在 MS 体绘制算法的基础上,研究和提出了一种体绘制中任意封闭多面体的剖切和多种变换函数,并进一步展现了数据场内部的数据分布情况. 另外,由于通过二叉树对多剖切体情况下 Stencil 参照值的合并,使得算法在每一切层上的绘制次数达到最少,同时还统一了剖切体前后表面 Stencil 操作,并减少了不必要的法线运算,从而大大加快了复杂剖切体的剖切操作.

关键词 可视化 体绘制 剖切 有限元网格
中图法分类号: TP391.72 **文献标识码**: A **文章编号**: 1006-8961(2002)01-0055-08

Cutting Algorithm in the Volume Rendering of FEM

YANG Xiao-song, GU Yuan-xian, LI Yun-peng, GUAN Zhen-qun
(*State Key Laboratory of Structure Analysis for Industry Equipment*, *Dept. of Engineering Mechanics*, *Dalian University of Technology*, *Dalian* 116024)

Abstract Volume rendering is a very effectived method to help user to find problems inside a very complicated structure. Compared with surface based visualization methods, it keeps more detail information in the final 2D image. This advantage just brings up another big problem. Usually some important information is obscured or overwhelmed by the data standing closer to the observer. To solve this problem, two algorithms are put forward in this paper. Cutting the structure with an arbitrary convex polyhedron to bring forward the focused part. By using the hardware supported stencil function in OpenGL, the cutting operation become very fast and easy. Another method is the definition of multiple transfer functions. The integration of these two methods makes the user's inspection more flexible. To achieve nearly real time response, several optimization methods are taken in this paper. The rendering time in each slice is decreased to the minimum due to the combination of stencil reference values by using binary tree. The integration of the stencil operations between the front and back faces of polyhedron speed the algorithm greatly by saving lots of complicate normal computations.

Keywords Visualization, Volume rendering, Cutting, Finite element meshes

0 引 言

体绘制与其他三维有限元数据场的可视化方法相比,其主要优点是能够在一个二维图中展现出更多的数据信息,但信息的增多,则不可避免地会造成图象信息之间的互相遮挡,为了解决该问题,虽然设置了透明度,但位于结构后面的单元往往在最终图中的贡献量非常低,甚至完全无法辨认,因此如何解决体绘制中遮挡问题,是近年来,体绘制研究中的一个重要问题.

通常有两种解决此问题方法,其中一种是用剖切方法(即通过剖切将前面的遮挡部分完全去掉)来展现结构后面的部分,如今体绘制中的剖切方法,依据剖切面的不同一般分为平面剖切[1]和任意封闭多面体剖切两类,其中,平面剖切由于计算比较简单,

基金项目:国家杰出青年科学基金项目(19525206);国家重点基础研究专项经费(G1999032805)
收稿日期:2000-06-09;改回日期:2000-11-20

因此对体绘制的绘制速度基本上不构成任何影响，而任意封闭多面体的剖切则比较复杂，如今提高体绘制效率的措施，通常都是采用硬件加速的策略，以提高体绘制的响应速度，例如 Westermann 利用 OpenGL 的三维纹理硬件加速技术实现了任意封闭多面体对体数据的剖切[2]；另一种方法是通过对数据场定义多个变换函数[3]来进行体数据剖切. 由于在体绘制中，变换函数直接影响着单元的色彩与透明度，因此通过定义多个变换函数，在不影响正常体光照模型的情况下，再通过降低数据集中某一部分单元的透明度，同样也可以达到展现数据场内部情况的效果.

本文综合利用了上述两类方法的优点，在 MS 体绘制方法[4]的基础上，利用任意封闭的剖切体与有限元模型进行相交的方法，来对数据场中的单元进行分类，并通过对不同的子集定义不同的映射变换函数，来为用户提供一种非常灵活的展现结构内部数据分布的方法. 由于采用硬件直接支持的 OpenGL 的 Stencil 功能来实现单元的剖切，其算法效率和响应速度很高，因此通过算法分析，本文在如下几个方面对算法进行了改进，从而使算法效率得到大幅度提高：

(1) 通过二叉树来合并多个剖切体情况的 Stencil 参照值，以使每一切层上的绘制次数达到最少；

(2) 利用包围盒技术来对 Stencil 测试和深度测试进行优化，以减少不必要的象素填充操作；

(3) 利用 OpenGL 的 Scissor 和包围盒技术，来减少外表面面片与剖切体的求交操作；

(4) 通过对剖切体前后表面 Stencil 操作的统一，从而节省了复杂剖切体绘制中繁琐的面片求法线运算.

1 体绘制中的剖切算法

本文算法是在 MS 算法的基础上，利用 Stencil 功能来实现三维有限元数据场体绘制的剖切操作，其算法的流程如下：

(1) 定义封闭多面体以及相应映射变换函数；

(2) for each slice do；

(3) 采用由封闭多面体与当前切面相切的方法来设置 Stencil 缓冲区的值；

(4) 处理边、单元、外表面的活性表；

(5) 绘制相邻剖切面之间的外表面；

(6) 形成剖切多边形；

(7) 绘制剖切面多边形，并累加.

1.1 任意封闭多面体与变换函数的定义

这里，多种变换函数与剖切，实际上就是在单元上定义了一种集合关系，若每一个封闭多面体都意味着对三维空间的一个二值划分，即将多面体划分为内外两个部分，则集合 P_j 的定义如下

$$P_j = \{E_i, E_{i+1}, \cdots\} \tag{1}$$

式中，P 代表封闭多面体，E 代表单元. 由该式可见，一个单元可以同时属于多个封闭多面体. 由于每一个集合都对应着一个变换函数，所以当一个单元属于多个集合时，该单元所采用的变换函数将取决于各集合之间的并、交、差操作.

变换函数的定义要根据数据场的物理意义和剖切多面体的用途而定. 其中对物理意义的数据结构，如应力场等，通常采用如下所示的二值函数：

$$F(s) = \begin{cases} \rho_{\text{high}} = 0.93 & \text{if} \quad s > T \\ \rho_{\text{low}} = 0.07 & \text{if} \quad s < T \end{cases} \tag{2}$$

式中，s 为应力值，ρ_{high} 为高阻光度，ρ_{low} 为低阻光度，T 为阈值. 对于剖切掉的部分则无需定义变换函数，在设置 Stencil 时，这类单元将不参加绘制操作.

1.2 OpenGL 中 stencil 功能

Stencil 是 OpenGL 中的一个重要功能，因为 Stencil 测试通常与深度测试共同来控制图形的绘制，且这种测试主要由 Stencil_Func 和 Stencil_Op 两个函数来完成，其中，Stencil_Func 用于决定通过 stencil 测试的区域是否进行绘制，而 Stencil_Op 则用于决定 stencil 测试的结果对 stencil 缓冲区本身的影响. Stencil 测试可以采取的操作有GL_KEEP、GL_ZERO、GL_REPLACE、GL_INCR、GL_DECR、GL_ INVERT 等. 本文算法则利用 GL_INCR 和 GL_DECR 来实现体绘制的剖切.

在 Stencil 操作中，由于 Stencil 缓冲区中的值起着至关重要的作用，因此本文利用 Stencil 缓冲区中的值对单元进行分类. 目前微机显卡大都支持 8bit 的 Stencil 缓冲区，即可以实现最多 8 个剖切多面体的定义. 如表 1 所示，若某象素(某一单元与当前切面相切所得的切面多边形中的象素)位于第 3、4 和 7 剖切多面体内部，则在绘制该象素时，就可以依据第 3、4 和 7 集合之间的集合操作，并采用相应的变换函数来将数据映射为颜色和透明度，以便参与体绘制的积分运算.

表 1

象素所对应的 Stencil 缓冲区内的值	按位表示							
50	0	0	1	1	0	0	1	0

1.3　利用 stencil 功能实现体绘制中的剖切算法

MS 算法是通过一组平行于投影面的平面与结构相切的方法,将各剖切面多边形依次累加来实现体绘制的. 图 1 给出了单元与当前切面相切时形成的切面多边形的示例,而体绘制的剖切,关键就是在填充切面多边形时,对填充象素进行分类,即判断哪些象素是位于剖切多面体内部的象素. 这样,三维体绘制中的剖切就转换成一个两维点集的分类问题.

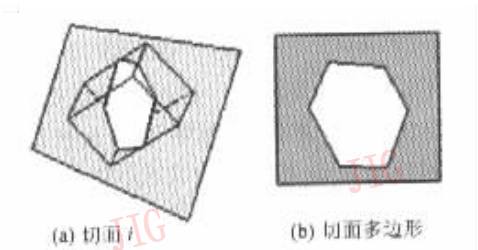

图 1　单元与切面相切形成的切面多边形

这里,首先以一个剖切多面体的情况为例,来讨论如何实现体绘制. 图 2(a)给出了一个简单的剖切体——球体. 在实现体绘制时,算法的目的就是要判断在当前切面上,哪些象素位于球体的内部,即对切面中的所有象素定义一个集合;图 2(b)给出了球体与当前切面的相交圆,这一相交圆实际上就定义了象素集合的划分方法,其中位于圆内的象素就是属于剖切多面体应该剖切掉的部分. 这样只要将 Stencil 缓冲区中相应的象素设置为 1,然后在绘制单元的切面多边形时,就可禁止相交圆内的象素绘制,这也就实现了体绘制的剖切.

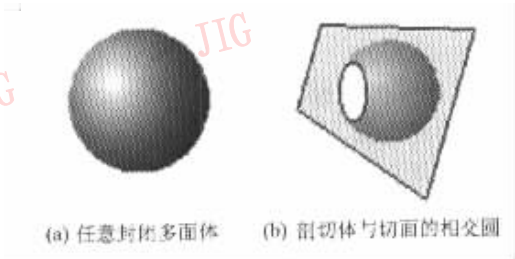

图 2　任意封闭多面体与切面相切

这里,相交圆的求法虽可以采用软件的方法,但由于每一个切面都要进行这种求交操作,因此运算量不是很高. 本文通过硬件支持的 Stencil 测试来求得相交圆,具体方法如下:

1.3.1　体绘制前的准备工作

(1) 封闭多面体前后面的划分,如图 3(a)所示. 首先将剖切多面体的各顶点坐标映射到屏幕坐标系,然后在屏幕坐标系内,求得剖切多面体各面片的法线方向,其法线方向向外,再求该矢量与单位向量(0,0,1)的点积,如果结果为正,则说明该面片方向向前,为前表面,否则为后表面.

(2) 打开 Stencil 测试开关,允许对 Stencil 缓冲区进行修改,然后关掉深度测试.

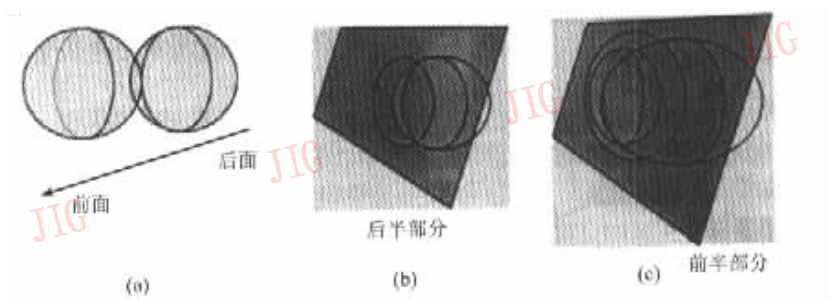

图 3　Stencil 缓冲区内容的设置

1.3.2　在体绘制循环内部对每一个切层进行以下操作:

(1) 清除 Stencil 缓冲区中的内容,将深度缓冲区中的内容置成当前切面的 Z 坐标值,然后打开深度测试开关,并关掉深度缓冲区与颜色缓冲区的写开关,即在设置 Stencil 缓冲区时,不允许修改已有的体绘制图象.

(2) Stencil_Func 中,测试函数将设置为 GL_ALWAYS,将 RefVal 置为 1,即 Stencil 测试的比较结果为通过,且在 Stencil_Op 中,与 Stencil 测试和深度测试都通过对应的操作,均采用GL_INCR,即增 1 操作,然后绘制剖切多面体的所有后表面,如图 3(b)所示,这时由于后表面完全位于当前切面的后面,故深度测试通过. 另外,由于 Stencil 测试为通过,所以在所有后表面所覆盖的象素上(相交圆的内部)Stencil 缓冲区的内容由 0 增加为 1,其余皆为 0.

(3) 不改变 Stencil_Func 的设置,即 Stencil 测试为永远通过,并且在 Stencil_Op 中,与深度测试和 Stencil 测试都通过对应的操作均采用 GL_DECR,即减 1 操作,然后绘制剖切多面体的所有前表面,如图 3(c)所示,这时通过深度测试的部分,在切面上即形成一个圆环,而且其圆环内部的象素在 Stencil 缓冲区中的值由 1 减为 0,小圆内部仍为 1. 其实这一部分为 1 的象素正是剖切多面体与当前切面的相交圆,也正是本文算法需要求得的.

(4) 关闭深度测试,打开颜色缓冲区写开关,Stencil_Op 将设置为 GL_KEEP,即不对 Stencil 缓冲区进行任何改变,同时将 Stencil_Func 中的 RefVal 置为 0x01,然后比较函数,若为不等于,则允许剖切多面体外面的部分参加绘制,并开始绘制单元的切面多边形、小单元与外表面.

1.4 多个剖切多面体的结合

无论采用多少个剖切多面体,其关键问题仍然是如何进行 2 维切面上象素集的划分问题. 如图 4 所示,n 个剖切多面体即意味着将象素集分为 2^n 个子集,且每个子集对应一个变换函数. 在一般情况下,有 3 个剖切多面体已经足够了,因为若定义 8 个以上变换函数,对于应用来说,实际上是非常不方便的.

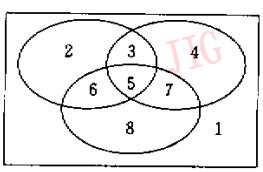

图4 3个剖切体的相交圆

本文将这种集合划分为如下两类:其中一类是进行纯粹的剖切,即位于剖切多面体内部的部分将完全剖切掉,而不参加体绘制的积分运算;另一类集合划分则是要为每一个剖切多面体定义一个变换函数. 这种变换函数由用户指定,通常采用的是如式 2 所示的高通函数. 由于阈值 T 的选取将决定该连通域的用途,因此 $T=1$ 可视为为了展现结构内部为透明的特殊情况,同时,高阻光度 ρ_{High}、低阻光度 ρ_{Low}和 T 3 个参数的灵活选取,为用户提供了探索结构内部数据分布的十分有效的方法.

由于每一个剖切体可以用 Stencil 缓冲区中的一个位来表示,因此对于一般的硬件配置来讲,最多可以支持 8 个剖切体. 由于不同的变换函数需要对单元顶点的颜色和透明度重新计算,且每增加一个变换函数就需要增加至少一次绘制操作,所以变换函数越少,绘制次数也就越少,算法的响应速度也就相应地越快. 由于每一次绘制将只对应一个固定的 Stencil 参照值,所以绘制次数直接等于 Stencil 参照值的数目. 图 5 给出了图 4 中各部分的 Stencil 参照值,如果 3 个剖切体都是纯粹剖切,那么是否需要按照 7 个参照值绘制 7 次呢?在这种情况下,实际上,只需要按照 7 个参照值合并的结果 000 绘制一次就可以了. 这样,减少参照值就成了提高算法效率的关键.

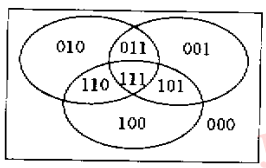

图5 3个剖切体中不同部分的Stencil比较的参照值

参照值的合并算法是一种按位的操作,如设 A 和 B 为要合并的参照值,则可将 A 和 B 按位进行异或操作,设结果为 C. 如果 C 中只有一位为 1,则 A 和 B 可以合并;如果全为 0,则 A 和 B 相同. 绘制时将 Stencil 比较的 Mask 设为 C 的反,参照值可以选择为 A 或 B. 以 010 和 011 为例来说明,其异或结果为 001,此时如果采用 110 作为掩码,那么只需要比较前两位,即只需比较 01 就可以了. 如果合并的参照值比较多,则需要将掩码和参照值一起传递下去. 此时被掩码遮掉的位可以用 X 表示. 现以 010、011、110 和 111 的合并为例,若前两个合并为 01X,后两个为 11X,再次合并为 X1X,则最终的掩码为 010. 但并不是所有的参照值都可以合并,例如 010 和 001 就无法进行合并,它们要进行两次绘制才能完成.

最终的合并结果实际上是取决于合并次序,如在合并 010、011、001、110、111 和 101 时,如果 010 和 011 合并为 01X,而 111 和 101 合并为 1X1,那么剩下的就无法合并下去了,最终的绘制次数为 4. 如果合并次序为 010+011=01X,110+111=11X,11X+01X=X1X,001+101=X01,则最终的绘制次数为 2,且时间将是前面的一半,所以合并次序对于提高算法效率是十分重要的.

解决的方法是利用二叉树来对参照值进行合并,且二叉树的每一层代表着一个剖切体的集合划分,但在建立二叉树之前,需要确定 3 个剖切体的顺序,即首先对所有要合并的参照值进行分析,统计每一个位上相同值的个数,然后哪一位的数目越高,其在树上的位置也就越高. 以上面的 6 个参照值为例进行分析可见,由于中间位有 4 个 1、2 个 0,最后一

位也有 4 个 1 和 2 个 0,只有第 1 位有 3 个 1 和 3 个 0,所以可以取中间位为树的第 1 层,最后一位为第 2 层,第 1 位为第 3 层. 这样 6 个参照值中,中间位为 1 的 4 个值就可以合并到一起,而不会因为位于不同的树的分支,而无法合并. 图 6 以上面的 3 个剖切体为例,给出了一个二叉树的结构,其叶节点为未合并前的参照值,中间的节点为合并后的参照值,树的深度为剖切体数目加一,每一个节点仅记录一个布尔数,即只记录该节点是否有值. 图 6 还给出了上面所述的 6 个参照值合并的例子,合并的最终结果为两个 X01 和 X1X,这是最优解.

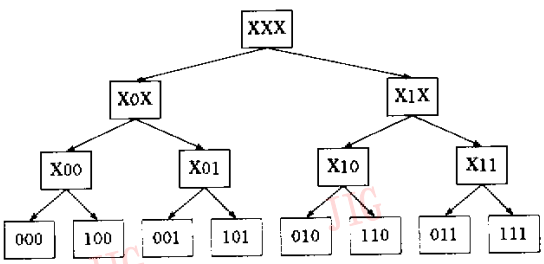

图6　6个参照值的合并实例

1.5　利用包围盒加速剖切操作

本文算法是建立在 MS 体绘制算法之上的,由于体绘制算法的关键问题是响应速度,所以必须对剖切操作的系统开销进行分析,以便在原算法的时间开销之外,实现最小的时间增长.

据分析,剖切操作涉及到以下几部分操作:

(1) 预处理. 这里仅涉及到剖切多面体的坐标变换等一次性操作,由于一般剖切多面体都比较简单,且顶点与面数都不是很多,因此这部分操作所消耗的时间基本上可以忽略不计.

(2) 在体绘制循环内部的操作. 由于对于每一个切面来讲,所有操作都要进行一遍,所以这里所消耗的时间就至关重要. 这里有两项比较消耗时间的操作,其一是清除 Stencil 缓冲区和深度缓冲区,另一个是剖切多面体前后表面的绘制. 同时这两项操作所消耗的时间将直接与显卡的象素填充速度相关,即如果填充速度比较慢,则这部分操作将成为剖切操作的瓶颈.

为了提高剖切操作的效率,本文采用包围盒的方法来减少填充象素的数目,即对于所有剖切多面体,首先在屏幕坐标系内,求其外包围盒(三维),然后为每个多面体记录 Z 轴方向的两个值 Z_{max}、Z_{min}. 通常情况下,由于多面体都是对结构的某一部分进行裁减,因此在 Z 轴方向贯穿的情况比较少,这样就避免了在 $Z<Z_{min}$ 和 $Z>Z_{max}$ 时,无谓的多面体前后表面的填充操作,若在 Z 轴方向多面体贯穿的层数越少,则求取包围盒节省的时间也就越多.

这样在每一个剖切面上,采用两维的包围盒技术,并利用 OpenGL 的 Scissor 函数在 Alpha 和 Stencil 测试之前,对填充区域进行裁减,就避免不必要的清 Stencil 缓冲区和深度缓冲区的操作,而且多面体在 X-Y 平面内覆盖的面积越少,算法效率提高得也就越大.

2　外表面面片的剖切

外表面的绘制作为 MS 算法的一部分,必须在剖切时加以考虑. 由于位于相邻切面之间的外表面面片不像切面那样具有统一的深度,因此处理将比较复杂. 如今进行外表面面片剖切所采用的方法可以有如下两种:

(1) 假设切面间的外表面面片具有统一的深度,这时可以采用与切面多边形绘制时,同样的 Stencil 缓冲区,且原有体绘制算法不变,但这是一种近似方法,其误差主要集中于剖切多面体与相邻切面的相交多边形的边缘处. 这样如果在体绘制中切面数越多,即相邻切面之间的间距越小,那么由此产生的误差也就越小,所以这种方法比较适合于切面数在 200 以上的情况.

(2) 当切面数比较少时,若采用第 1 种方法,由于在剖切多面体的边缘处,将出现明显的锯齿状的面片,因此这时应该采用比较精确的绘制方法,其需要改变的步骤主要是深度缓冲区的统一赋值,而且对于每一个外表面面片,应作如下操作:

步骤 1　清 Stencil 缓冲区,关闭颜色缓冲区的写开关;

步骤 2　绘制外表面面片,用于将面片的深度写入深度缓冲区;

步骤 3　同前面算法:分别绘制剖切多面体的前后表面,设置 Stencil 缓冲区;

步骤 4　关闭深度测试和深度缓冲区写开关,打开颜色缓冲区写开关,绘制外表面面片.

由于每一个面片都需要对 Stencil 缓冲区进行单独设置,因此算法所需要的时间要增长很多.

对此,本文主要采用下面的两种方法来提高算法的效率:

(1) 将面片的包围盒与多面体的包围盒进行简单的相交比较,如果面片位于多面体包围盒之外,则不必进行 Stencil 设置,而直接进行绘制即可,而且这种测试将去除掉大部分的面片.

(2) 如果第 1 种测试失败,则将采用前述的方法进行绘制,但由于面片一般都比较小,涉及的象素不是很多,这时若采用 OpenGL 的 Scissor 功能,就可以避免很多无谓的象素填充,而只影响到面片涉及的很小一部分区域.

实践证明,采用这两种方法可以节省很多时间.另外,由于切面数较少时,体绘制消耗的时间也比较少,所以从整体的时间消耗来讲,外表面面片的绘制对体绘制算法的影响不是很大.

3 实例分析

图 7 给出了一个由 6 300 个节点组成的实体结构,在一个长方体剖切情况下的体绘制图象. 为了清晰地观察到剖切面的情况,这里若采用纯粹剖切方法,结构其他部分的透明度则比较低,为 3%. 图 8 给出了该剖切结果图象,其剖切部分采用 93%的透明度进行绘制.

图 9 给出了该结构在两个剖切长方体剖切情况下的图象,其两个长方体都采用纯粹剖切方法.

图 10 在同样剖切情况下,其中一长方体采用纯粹的剖切,另一个采用比较高的透明度.

从图 11 中可以看出,两个长方体将结构中所有单元分成 4 个集合,其中,(2)和(4)其各部分如图 11 所示,同属一个集合. 如果一长方体采用 Stencil 缓冲区中最右边一位,另一个长方体采用第 2 位,则(1)~(5)各部分的 Stencil 参照值分别为 0X11、0X10、0X01、0X10、0X00;如果两者都采用纯粹剖切(图 9),则可以将 Stencil 参照值设为 0X00. 图 10 的绘制则需要两步操作,即首先如图 9 一样,绘制第(5)部分,然后将 Stencil 参照值设为 0X10,透明度设为 93%,再对(2)(4)部分进行绘制即可.

图 7 单一长方体的纯粹剖切结果

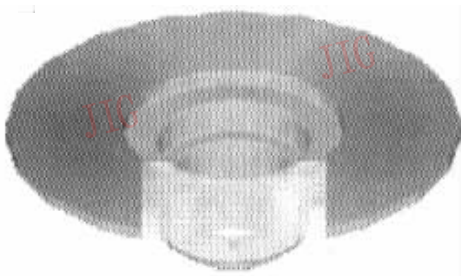

图 8 单一长方体的剖切结果(透明度 93%)

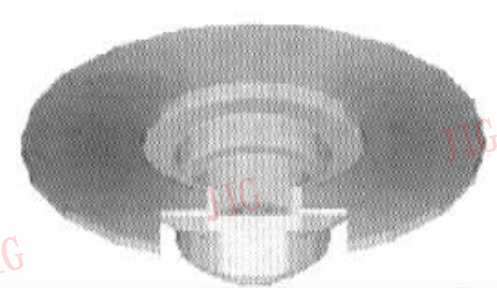

图 9 两个长方体对结构的纯粹剖切结果

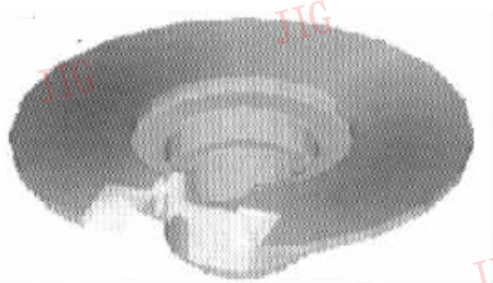

图 10 两个长方体剖切的透明度控制

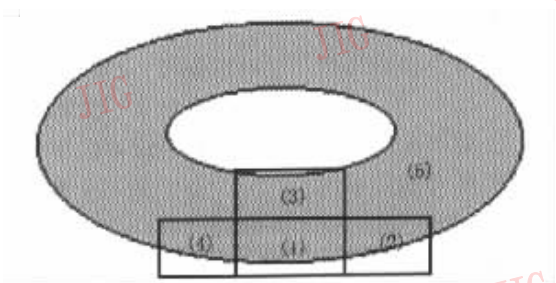

图 11 两个长方体的剖切示意图

4　复杂剖切体的实现

大家知道，长方体是比较常用的剖切体，由于其结构比较简单，因此往往无法满足用户的要求. 除了长方体外，通常采用的剖切体还有球体、圆柱以及圆锥等等，但构成这类剖切体的表面已经不是平面，而是曲面. 为了处理方便和提高算法效率，通常是采用小面片来进行逼近. 为了达到比较好的效果，面片数量可以选取得比较多，因而这就为剖切带来了一定的难度，因为当视角改变时，由于剖切体的前后表面需要重新判定，因而这也就涉及到所有面片法线方向的求解，由于运算量相当大，所以必须对算法进行改进，以便进一步减少前后表面的判定运算.

通过对算法的分析可以发现，之所以要进行剖切体前后表面的判定，其主要原因是绘制过程不一致，因为在绘制后表面时，Stencil 操作是增 1，而绘制前表面时，Stencil 操作是减 1. 如果能够将两者统一起来，就不需要进行前后表面的判定，也就可以节省掉剖切中繁重的法线计算，但问题是如何选择合适的 Stencil 操作，以使得对每个剖切体所对应的位来讲，两次同样操作数据不变，即该操作的二值运算如下：

$$Op(0)=1, Op(1)=0$$

此时，可以采用的操作虽然有 GL_INCR、GL_DECR 和 GL_INVERT 3 种. 然而由于 GL_INCR 和 GL_DECR 在进位和借位时，将改变其他位的值，这将影响到其他剖切体的集合定义，所以只有 GL_INVERT 能够正确实现这一操作.

这样由于前后表面绘制操作的统一，因此在预处理时，就不需要进行前后表面的判定，而只要将剖切体的各个面片依次绘制即可，而且即使对于比较复杂的封闭剖切体，其算法效率仍然比较高. 图 12 和 13 给出了将球体和圆台作为剖切体的情况.

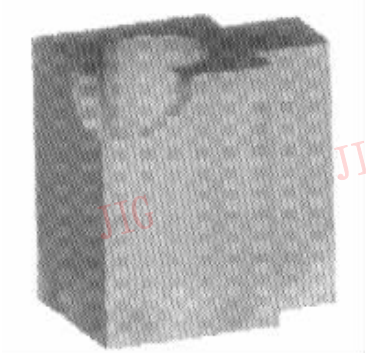

图 12　球的剖切

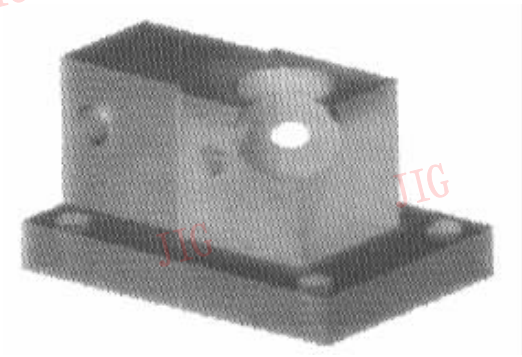

图 13　圆台的剖切

5　结　论

本文在有限元网格 MS 体绘制算法的基础上，提出了一种多个封闭多面体的剖切算法，并给出了多种变换函数的定义，从而为用户提供了一个十分灵活的观察数据场内部情况的方法.

由于本文通过二叉树方法对 Stencil 参照值进行了合并，因而大大减少了在每一切层上绘制的次数，并使得算法效率得到了很大的提高. 下表给出了图 11 中合并后的参照值的数目与绘制时间的对比（300 切层）.

表 2　图 11 合并后的参照值数目与绘制时间对照表

参照值数	1	2	3	4
绘制时间(s)	11.374	13.609	15.16	16.47

对于复杂的剖切体，通过对其前后面的合并，使得算法在设置 Stencil 操作上，效率提高了近 10%，以图 12 为例，在原算法中，是利用 OpenGL 的 glCullFace 功能来判断剖切球的前后表面，其 Stencil 缓冲区的设置时间为 3.23s（300 切层），而采用 GL_INVERT 合并操作后的时间为 2.81s.

另外，由于可利用包围盒和 OpenGL 的裁减功能来减少不必要的 Stencil 测试，从而可以大幅度地提高算法的效率，因为 Stencil 测试所消耗时间往往比较大，如图 12 所示结构，在无剖切时，其绘制的时间为 3.645s（300 切层），利用球体剖切的绘制时间为 9.934s，而采用包围盒技术后的绘制时间为 4.998s. 另外，算法效率提高的程度还和剖切体与结构的相交程度有关，即相交的部分越小，算法效率提高得也就越大.

参 考 文 献

1　Mark W. Jones, Min Chen. Fast cutting operations on three di-

mensional volume datasets[A]. In:edi Visualization in Scientific Computing[M], New York: Springer-Verlag, January 1995: 1～8.
2 Westermann R, Ertl T. Efficiently using graphics hardware in volume rendering applications[A]. In:Proceedings of Computer Graphics Annual Conference Series, ACM Press, 1998:169～177.
3 Michael Meißner, Ulrich Hoffmann, Wolfgang Straßer. Enabling classification and shading for 3D texture mapping based volume rendering using OpenGL and extensions[A]. In: Proceedings of the Conference on Visualization '99: Celebrating ten years[C], October 24-29, 1999, San Francisco, CA, USA. 1999:207～214.
4 杨晓松. 三维有限元数据场可视化技术研究与实现[D]. 大连:大连理工大学,2000.

杨晓松 1971 年生,2000 年获大连理工大学博士学位,现在清华大学从事博士后研究. 研究方向为可视化与图形学.

顾元宪 教授,博士生导师. 研究方向为计算力学与 CAD.

李云鹏 讲师,研究方向为有限元及其前后处理.

关振群 讲师,研究方向为 CAD 及有限元前后处理.

影象信息 轻松传递
——虹软科技(北京)公司成立

手机、电脑、互联网、PDA、数码相机已日益成为人们传递信息的新宠,并逐步成为信息社会的主导,以图形图象为核心的多媒体信息的传递将成为最终发展方向,而市场对其发展呈现出了迫切的需求. 伴随着这一趋势,作为全球数码影象处理领域的领导企业,虹软科技(北京)公司于近日正式成立,并推出了多个先进的影象处理软件,以及面向互联网与无线网络上的影象存储、处理及传输的软件解决方案,以期完善在网络时代下影象信息与传输系统的完美结合.

此次紧随公司的成立,虹软推出了最新的两个数码信息的处理软件. 其中,ArcSoft PhotoImpression 是一个有趣的入门级的数字图象编辑应用软件,是可利用的最好的比较全面的图象编辑器. 该产品提供了大量用于改进图象质量的功能,具有完整的编辑,润饰和增强工具. 友好的用户界面适合于那些对简单易用的图象编辑解决方案的初级用户的需求. ArcSoft ShowBiz™是虹软公司最新推出的顶级视频产品,是一个完整的电影制作和编辑系统. ShowBiz 给用户提供了各种工具来生成家庭或商业的电影,它具有用户创建视频文档所需的所有功能,并可通过电子邮件,Internet 或存储在 CD-ROM 上进行分发. 它支持多种基于 PC 的数字视频格式和压缩技术,这些先进的功能还将改进用户的播放效果. ShowBiz 是虹软公司特意给初学者和电视迷们开发的,初学者可以快速制作视频,而高级用户从中可以发现更多的功能.

随着宽带技术和无线网络接入设备的发展及应用推广,多媒体技术正成为移动互联网应用发展的重要方向. 基于多年在数码影象信息领域的研究和应用,虹软公司在为全球知名企业提供移动互联网图象技术解决方案的过程中积累了丰富的经验,率先推出了以虹软 M3 平台(Multi Media Management)为核心的移动互联图象整体解决方案. 该解决方案包括虹软 M3 网络多媒体共享系统、虹软 M3 信息传送服务系统、虹软 M3 终端应用套件. 它最大程度地解决了数码影象信息在无线网络上的传递交流,具有广阔的应用前景. 另外,手机制造商、无线互联网内容提供商、2.5G 和 3G 无线网络运营商都会受益于此系统,用户也将从中充分感受网络时代信息传递的便捷与内容的丰富.