基于切割距离场的体数据切割算法研究

谢 凯[1,2] 孙 刚[3] 杨 胜[1] 余厚全[2] 杨 杰[3]
(湖南大学计算机与通信学院 长沙 410082)[1] (长江大学电子信息学院 荆州 434023)[2]
(上海交通大学图像处理与模式识别研究所 上海 200240)[3]

摘 要 为了精确地对体数据进行可以交互的切割模拟,提出了切割距离场的概念和一种基于切割距离场的三维体数据切割算法,该算法利用切割距离场作为切割体的数据表示形式,在图形处理单元(GPU)的着色器中同时实现体重建和实时切割两种功能,实验结果表明该切割算法可以达到可交互的速度,可以处理任意形状的切割体。

关键词 医学图像,三维切割模拟,切割距离场,深度缓存,在图形处理单元
中图法分类号 TP391 **文献标识码** A

Volume Clipping Simulation Based on Clipping Distance Field

XIE Kai[1,2] SUN Gang[3] YANG Sheng[1] YU Hou-quan[2] YANG Jie[3]
(School of Computer and Communication, Hunan University, Changsha 410082, China)[1]
(School of Electronic Information, Yangtze River University, Jingzhou 434023, China)[2]
(Inst. of Image Processing & Pattern Recognition, Shanghai Jiaotong University, Shanghai 200240, China)[3]

Abstract In order to clip volume data by interactive speed, we proposed the concept of Clipping Distance Field (CDF) and a CDF based clipping simulation algorithm. By combining volume rendering and clipping in the shader of GPU, the algorithm also achieved the high rendering/clipping speed required by interactive simulation. The result shows that the algorithm can process clipped data with different shape.

Keywords Medical image, 3D Clipping simulation, Clipping distance field, Depth buffer

1 引言

体数据切割[1]在医学图像理解、辅助诊断、虚拟手术及虚拟雕塑中起着至关重要的作用。一方面,将数据切开可以更加清晰地查看内部的重要细节;另一方面,手术模拟也必须实现切割操作。由于人体中有很多具有对称结构的组织,精确地对称切割有很多重要的应用。例如在脸部整形手术中,为了保证术后美观,必须保证切割是精确对称的,因此对称切割在手术模拟中有重要的地位。本文尝试采用图形显示卡的顶点着色器和像素着色器功能加速体数据的实时交互切割,将切割几何体用离散距离场[2]表示,在像素着色器中利用距离场来判断是否满足切割条件。

2 基于切割距离场的体数据切割算法

2.1 通用算法描述

首先讨论利用深度缓存来实现对体数据的切割模拟,假设切割物体是由其边界的表面表示。用于切割的表面应该是无缝的,通常用三角面片表示。图1显示了一个典型的切割体的深度结构。在下面的叙述过程中,将不再考虑三维场景,而是直接将问题简化为一条光线从眼睛到场景的一维几何问题。从视点出发,只考虑图像平面上的一个单独的像素。因此,以下的描述可以把所有的操作映射到每个独立像素所对应的片元上(像素着色器)。

我们从切割物体为任意拓扑结构或者几何形状开始说明一个通用的算法。首先,为当前的像素建立切割体的深度结构。这个深度结构保存着实现与切割体的每个相交点的深度值。运用这种方式,一维深度值被分割为了很多段,每一段的边界由切割体的边界给出。并且,将分割出来的间隔分类为在切割体的内部和在切割体的外部。分类结果由原始切割体决定,并且与无缝曲面的方向相关。对于一个闭合的,非自相交的切割表面,内部和外部的分类结果将交互出现在分割处理的段上。用户可以选在切割掉切割体的内部或者切割掉切割体的外部。这样,分割出的间隔也就唯一地表示了可见部分和不可见部分。其次,一个体数据切片的每个片元的绘制必须检查该片元属于哪一类间隔。最后,根据可见性,片元被混合到现实缓存中或者被切割掉。

到稿日期:2009-04-15 返修日期:2009-07-01 本文受高等学校博士学科点专项科研基金(20070532077),湖北省教育厅科研基金(Q200912 11)资助。

谢 凯(1974-),男,博士,副教授,主要研究方向为图像处理等,E-mail:kaixie2007@yahoo.com;孙 刚(1982-),男,硕士,主要研究方向为图像处理等;杨 胜(1977-),男,副教授,主要研究方向为图像处理等;余厚全(1958-),男,教授,主要研究方向为图像处理与通信;杨 杰(1963-),男,教授,主要研究方向为图像处理和模式识别。

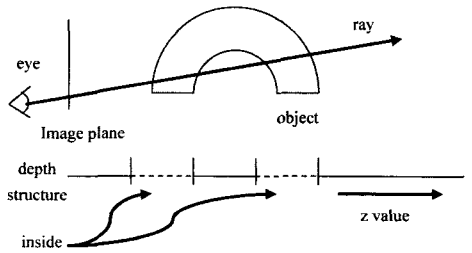

图 1　深度缓存示意图

2.2　GPU 加速的基于深度缓存的切割算法

实现立体切割的关键是判断图元是否在切割体内部，该方法使用深度缓存完成这个判断，并用图形处理单元加速这一判断过程。根据几何学，凸体内部任意点的连线都在凸体内部，即凸体与任意直线至多有两个交点，因此凸体切割很容易用深度测试表示；而凹体不具有这样的性质，因此本算法不适用于凹体切割。图 2 给出了该切割算法的二维示意图。

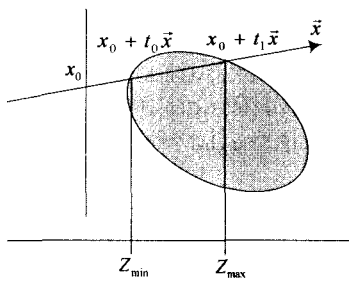

图 2　基于深度缓存的切割示意图

从眼睛(透视投影)或者无穷远(正交投影)发射一条射线，假设射线方向为 $\vec{x}$，与视平面的交点为 x_0，假设射线上一点 P 到该交点的距离为 t，则射线上的任意一点可以表示为 $x_0+t\vec{x}$，其对应的深度值是 t 的函数，可以表示为 $Z(t)$，其中 $Z(*)$ 为线性变换。假设射线与切割凸体相交并且有两个焦点 $x_0+t_0\vec{x}$ 和 $x_0+t_1\vec{x}$，并知道 $Z(t_0)=Z_{min}$ 和 $Z(t_1)=Z_{max}$。因此凸体内部可以表示为集合 U_{in}：

$$U_{in}=\{P|P=x_0+t\vec{x},Z(t)\leqslant Z_{max}\}\cap\{P|P=x_0+t\vec{x},Z(t)\geqslant Z_{min}\} \quad (1)$$

凸体的外部可以表示为集合 U_{out}：

$$U_{out}=\{P|P=x_0+t\vec{x},Z(t)\geqslant Z_{max}\}\cup\{P|P=x_0+t\vec{x},Z(t)\leqslant Z_{min}\} \quad (2)$$

可以证明 $U_{in}\cap U_{out}=\Phi$ 并且 $U_{in}\cup U_{out}=\Omega$。

因此，在执行体重建算法[3,4]时可以对每个图元进行深度比较操作，以此判断图元应该被切除或保留。下面给出详细的算法实现过程。

第一步，关闭图形显示卡的颜色缓存输出，打开深度测试功能。将深度缓存设为 0.0，将深度测试函数设为 GL_GREATER；然后将深度缓存拷贝在一个二维纹理中，假设为 BackTexture。设置深度缓存为 1.0，将深度测试函数设为 GL_LESS，绘制切割体的正面，然后将深度缓存拷贝在一个二维纹理中，假设为 FrontTexture。

第二步，打开颜色缓存输出，关闭深度测试，初始化像素着色器。在像素着色器中，利用图元在屏幕中的位置在 BackTexture 纹理和 FrontTexture 纹理中查找该图元对应的 Z_{min} 和 Z_{max}，并与当前图元的深度值比较。如果需要凸体内的部分，那么只保留深度值在 Z_{min} 和 Z_{max} 之间的图元；否则，只保留深度值在 Z_{min} 和 Z_{max} 之外的图元。

基于深度缓存的切割方法能够有效地处理切割体为凸体的切割问题，但是如果切割体为凹体，那么这种方法就不能完全适用了。准确地说，只有当所有视线与切割体相交次数少于或等于两次的时候，基于深度缓存的切割方法才能得到正确的切割效果。图 3 为深度缓存切割方法发生错误的示意图，其中黑色粗线为 BackTexture 和 FrontTexture 保存的深度值，因此深度缓存切割方法仅显示白色部分，但是正确的显示结果应该包括白色部分和灰色网格部分，在实验结果部分还将给出深度缓存切割方法发生错误的示意图和效果图。

图 3　基于深度缓存的切割模拟方法出错示意图

2.3　基于切割距离场的体数据切割算法

任意表面 S 可以用有符号距离场 D 表示，有符号距离场 D 定义如下：

$$D:\mathbb{R}^3\rightarrow\mathbb{R} \quad \forall p\in D$$

$$D(p)=\mathrm{sgn}(p)*\min\{\|p-q\|:q\in S\} \quad (3)$$

其中，

$$\mathrm{sgn}(p)=\begin{cases}-1 & \text{if } p \text{ inside}\\ +1 & \text{if } p \text{ outside}\end{cases}$$

$\|*\|$ 表示距离。

对距离场进行三维均匀采样，得到离散距离场。该切割方法的思想是将离散距离场作为硬件纹理单元，然后在像素作色器中分别对三维体数据纹理和离散距离场纹理进行三线性差值采样，根据离散距离场的采样值判断该图元是否应该被切掉或保留。具体算法步骤如下：

1. 选择合适的切割体。

2. 根据选择的切割体，计算一个切割距离场，切割距离场经过采样后形成一个三维数组，数组的维数应为 2^n，以满足纹理硬件的要求。存储切割距离场的三维数组维数并不一定与表示三维体数据的纹理维数相同，但是切割距离场三维数组的数据维数越大，切割距离场表示就越精确，切割也就越精确。

3. 将表示切割距离场的三维数组归一化为[0,255]的整数，以符合硬件纹理的特点，归一化以后，切割距离场中为零的点对应于归一化后三维数组中值为 128 的点。下面给出一个参考的归一化方式：

$$texel(p)=\lfloor clamp(D(p)+128)+0.5\rfloor \quad (4)$$

其中，$texel(p)$ 表示对应于切割距离场 p 点的三维数组的值，$\lfloor x\rfloor$ 表示不超过 x 的最大整数：

$$clamp(x)=\begin{cases}0 & x<0\\ x & 0\leqslant x\leqslant 255\\ 255 & x>255\end{cases} \quad (5)$$

注意：在图形硬件中，距离场纹理还将进一步归一化到[0,1]之间。

4. 将表示切割距离场的三维数组作为纹理传输到图形硬

件。

5. 对原始体数据进行体重建。绘制时，每一个绘制代理多边形的每个顶点都赋予两个纹理坐标，一个纹理坐标对应于原始的体数据纹理，另一个纹理坐标对应于切割距离场对应的三维纹理。

6. 在图形处理单元(GPU)中，定点着色器和像素着色器分别利用切割距离场纹理坐标在切割距离场纹理中进行采样。如果采样结果大于 0.5，那么说明这个片元位于切割体的外部；如果采样结果小于 0.5，那么说明这个片元位于切割体的内部。然后根据需要，把要切除部分的不透明度设为 0，即不现实切除的部分；对于需要保留的部分，还是按照正常的体重建算法进行采样、分类、混合操作，然后显示到屏幕上。

3 实验结果及分析

3.1 基于切割距离场的切割模拟

本文算法基于 C++，OpenGL[5]，GLUT 和 Cg[6] 语言，是平台无关的实现，可以在 Linux 和 Window 上运行。下面的性能结果在一台 Window XP 个人电脑上测量得到，CPU 为 Intel Pentium 4 2.0GHz，图形处理单元(GPU)为 GeForce 6600(128M 纹理内存)。三维医学数据的维数为 256×256×84，视窗面积为 512^2 和 1024^2。

图 4 是根据本文算法实现的切割效果图，图 4(a)为使用一个立方体作为切割体的切割效果图，图 4(b)是采样球体作为切割体的切割效果图，图 4(c)采样了一个非常复杂的形体作为切割体的切割效果图。由于体数据有着更加丰富的内部细节，从图 4 中可以看出，相较于基于表面重建的切割方法，基于切割距离场的体数据切割方法有着一定的优势。

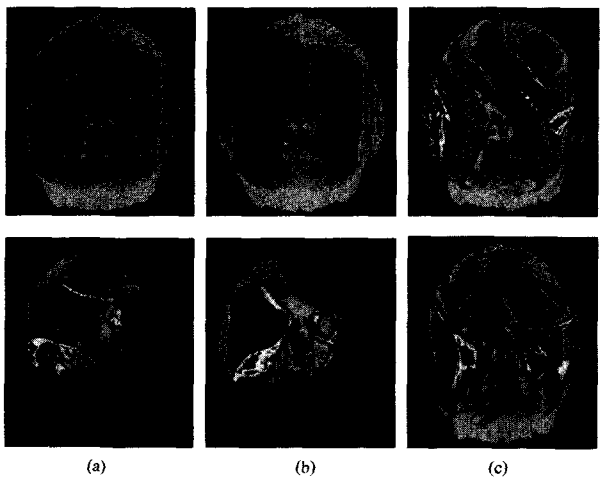

图 4 基于切割距离场的切割模拟

图 5 说明基于切割距离场的切割方法能够克服基于深度缓存的切割方法的缺陷，当切割体是非凸体且有自遮挡现象时，基于深度缓存的方法无法取得正确的结果，而基于切割距离场的方法能够克服这种缺陷，很好地生成结果图像。另外，由于基于表面重建的切割方法需要建立切割体所对应的隐函数，在切割体十分复杂的情况下，即使使用隐函数组合也很难精确地描述一个三维切割体，而基于切割距离场的切割方法采用体数据描述切割体，理论上无论切割体的几何形态、拓扑结构多么复杂，切割距离场都可以用相同的有限的数据量加以描述。因此，基于切割距离场的体数据切割方法是一种灵活的、有效的切割模拟方法。

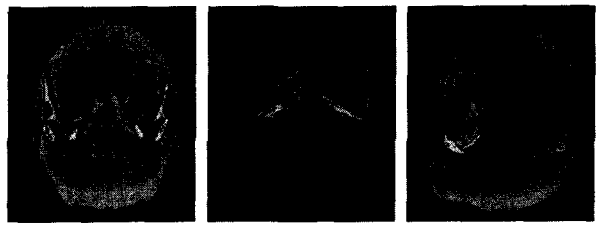

图 5 基于切割距离场的切割算法没有基于深度缓存的切割算法的局限性

3.2 不同切割操作性能对比

从表 1 中可以看出，切割效果的实现会引起交互性能的下降，一方面，纹理采样操作是一个非常消耗计算能力的操作，另一方面，相较于无切割效果的体重建，基于切割距离场的单物体切割多了一次三维纹理采样操作，因此在速度上有所下降。基于深度缓存的单物体切割，需要 3 个绘制过程，并且会有两个二维纹理的传输和采样操作，因此速度上有进一步的降低，但是由于图形硬件对于二维纹理操作有一定的优化，因此速度的降低并不十分明显。

表 1 各种切割操作性能比较表

视窗面积	512^2	1024^2
体重建(无切割效果)	9.68 FPS	2.9 FPS
基于深度缓存的单物体切割	5.68 FPS	1.45 FPS

结束语 本文的主要贡献表现在以下两个方面：一，提出了基于深度缓存的体数据三维切割模拟方法，该方法利用了图形硬件的着色器功能，在图形硬件中评估片元是否应该保留，提高了三维切割的速度，达到了可交互的速度，但是该方法有着很大的局限性，它只能处理切割体只有一个且为凸体的情况。二提出了切割距离场的概念和基于切割距离场的三维切割算法，利用 GPU 加速，该算法也达到了可交互的速度，并且该算法没有基于深度缓存的切割模拟算法的应用限制，可以处理任意形状的切割体。以后工作的重点是以基于切割距离场的三维切割算法为基础，研究多物体切割与对称切割算法，这两种算法有很大的实际应用价值，可以进一步丰富切割模拟算法并使其更加贴近实际应用。

参 考 文 献

[1] Weiskopf D, Engel K, Ertl T. Interactive clipping techniques for texture based volume visualization and volume shading[J]. IEEE Trans. Visual Comput. Graphics, 2003, 9(3)

[2] Gibson S F F. Using distance maps for accurate surface reconstruction in sampled volumes[C]// Volume Visualization Symposium'1998. IEEE, 1998: 23-30

[3] Levoy M. Efficient ray tracing of volume data[J]. ACM Trans. Graphics, 1990, 9(3): 245-261

[4] Cabral B, Cam N, Foran J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware[C]// Proceedings of the IEEE Symposium on Volume Visualization. 1994: 91-98

[5] Kessenich J, Baldwin D, Rost R. The OpenGL shading language, 2004[OL]. http://www.opengl.org

[6] Fernando R, Kilgard M. The Cg Tutorial-The Definitive Guide to Programmable Real-Time Graphics[M]. Addison Wesley, Reading, MA, 2003