第33卷 第20期 计 算 机 工 程 2007年10月
Vol.33 No.20 Computer Engineering October 2007
·多媒体技术及应用· 文章编号：1000—3428(2007)20—0222—03 文献标识码：A 中图分类号：TP393

基于图形处理器的近似等值面提取算法

郭 勇，顾力栩
(上海交通大学电子信息与电气工程学院计算机科学与工程系，上海 200240)

摘 要：提出了一种新颖的等值面提取算法，该算法基于Ray-Isosurface intersection方法，通过模拟三维扫描仪的工作过程来提取等值面。算法应用GPU的并行计算能力来完成主要的计算密度大的计算过程，计算结果存储为点的形式，并通过iso-splatting的技术绘制出来。算法通过控制GPU所使用的缓冲区大小和通过模拟三维扫描仪工作原理进而避免不可见部分的等值面提取，来达到一个高质量的绘制效果和高FPS的交互环境。通过该算法和传统算法在几组数据上面的绘制质量，FPS的比较证明了算法的优越性。

关键词：图形处理器；等值面提取；帧率

Isosurface Approximate Extraction Based on GPU

GUO Yong, GU Li-xu
(Department of Computer Science, School of Electronic Information and Electric Engineering, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 This paper proposes a novel isosurface extraction approach. This method is based on the algorithm of ray-isosurface intersection, simulating the 3D scanner to compute the isosurface. The proposed method employs the high parallel computing ability of the modern GPU to complete the main process, and the final result is represented as a set of points based on the method of iso-splatting. This approach achieves a high quality image and a high FPS rate by controlling the quantity of points and avoiding the invisible portions. To validate the approach against the traditional one, several datasets are tested in the experiment, and significant improvements are revealed in both FPS and quality.

【Key words】 graphics processing unit(GPU); isosurface extraction; frame per second (FPS)

等值面提取技术在科学数据可视化的领域内，特别是医学应用方面被广泛地使用。

传统的提取方法主要分为视角相关(view dependent)和视角无关(view independent)两大类。视角相关的代表性算法以Ray-Isosurface intersection[1]为代表，此方法的优点是可以生成解析度较高的等值面，但每次改变视角参数时都须重新计算，所以，计算量较大。

视角无关的提取算法的典型代表是MC(marching cubes)算法[2]。此算法和其改进算法[3]通过遍历整个体数据场，求出每个体素内所包含的近似等值面，使用三角片来表示等值面，也有使用点为基本图形单元的方法[4]。这类算法的主要特点是一次性提取等值面，当参数改变时无需重新计算，除非改变等值面大小。但是计算量大，特别是MC及其派生算法生成的三角片数量巨大，使用OpenGL或者Direct3D等图形API渲染起来比较缓慢，不利于实时交互。

本算法则通过利用GPU的并行计算能力，及其硬件执行三线性插值和高速的GPU－显存通信能力完成计算密度最大的部分，从而提高速度和精度[5]。通过模拟三维模型扫描仪(3D Scanner)的工作原理，从预先计算好的起点出发，使用基于Ray-Isosurface intersection的算法对三维数据场进行扫描，并使用微软的 Direct3D进行绘制。

1 Ray-Isosurface intersection 的概述

假设体数据是由规整的小格（体素）组成，那么Ray-Isosurface intersection算法实现起来比较简单。为了找到一个交点、一条射线(Ray) $\bar{a}+t\cdot\bar{b}$ 穿越体数据，用式(1)来计算出 t 的值。

$$\rho(x_a+t\cdot x_b, y_a+t\cdot y_b, z_a+t\cdot z_b)-\rho_{iso}=0 \tag{1}$$

2 算法实现

2.1 算法的数学基础

引理 1 体数据可以用三维笛卡儿坐标系下的一个长方体来表示：长，宽，高分别平行于3个坐标轴，左下角顶点坐标（V_x,V_y,V_z）长宽高方向上像素个数 X, Y, Z 和相邻像素间隔 S_x,S_y,S_z。

引理 2 对体数据内的一点，其对应的三维纹理坐标TexCoord的3个分量分别是世界坐标（笛卡儿坐标）x, y, z 的线性函数（参数定义见引理1），见式(2)~式(4)[6]：

$$T_{TexCoordx}=\frac{(x-V_x)}{X\cdot S_x} \tag{2}$$

$$T_{TexCoordy}=\frac{(y-V_y)}{Y\cdot S_y} \tag{3}$$

$$T_{TexCoordz}=\frac{(z-V_z)}{Z\cdot S_z} \tag{4}$$

定理 1 对体数据做Ray-Isosurface intersection，设世界坐标系下的光线变化方程为 $R(t)=B+\bar{I}\cdot t(0\le t\le L)$，$B$为Ray起始位置的世界坐标，$\bar{I}$ 为Ray前进方向，t为Ray步进次数，等价的世界坐标到纹理坐标变化方程如式(5)~式(7)所示[6]。$TR(t)$ 为纹理坐标（参数见引理1，引理2）。

$$TR_x(t)=\frac{I_x}{X\cdot S_x}\cdot t+\frac{B_x-V_x}{X\cdot S_x} \tag{5}$$

$$TR_y(t)=\frac{I_y}{Y\cdot S_y}\cdot t+\frac{B_y-V_y}{Y\cdot S_y} \tag{6}$$

作者简介：郭 勇(1981－)，男，硕士研究生，主研方向：GPU通用计算，科学可视化；顾力栩，博士、教授

收稿日期：2006-10-30 **E-mail：**apple_i@sogou.com

$$TR_z(t)=\frac{I_z}{Z\cdot S_z}\cdot t+\frac{B_z-V_z}{Z\cdot S_z} \tag{7}$$

本文算法使用以上公式，从预先计算好的起点开始，在GPU 内使用 Ray-Isosurface intersection 的算法，使用固定增量，来求取交点的纹理坐标，通过以上公式反向转换为世界坐标，然后将其存储在二维纹理的形式输出给 CPU。

2.2 扫描方式的确定

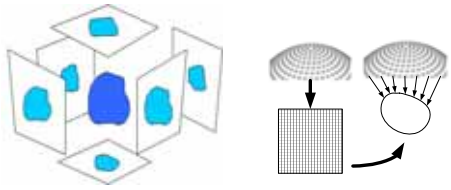
算法通过模拟三维模型扫描仪的工作原理对体数据进行等值面提取。由于三维模型扫描仪的扫描过程由人工控制，可以扫描到一些比较隐蔽的部位。但是本算法的等值面提取过程没有人的实时交互，因此提前确定好扫描的方式非常重要，否则提取的等值面将不理想。

正交扫描算法和球面扫描算法如下：

(1)正交扫描算法是从包围体数据的正方体的 6 个面分别进行垂直扫描。对体数据正前方的那个面而言，Ray从这个面上的每一点开始以垂直于这个面并朝着正后方进行扫描。只需要简单地计算这 6 个面的Ray的起点，然后存入 6 个二维纹理里面供GPU读取。例如存储正前方起始点的二维纹理，设置其大小为 1024·1024（可根据实际需要的精细程度而定，纹理越大则扫描起始点越多，从而增大扫描密度)，那么就有2^{20}个扫描起点，可以依次从左至右，从上至下地用起始点的世界坐标填充这些纹理像素RGBA值。但是这种扫描方式存在点重复的问题。

(2)球面扫描算法则相对复杂，首先通过球面公式生成均匀的点，然后将点存储为二维纹理供 GPU 使用。Ray 将由球面上的点出发，向着球心的方向前进。球面扫描可以完全消除点的重复问题，同时只需一个二维纹理即可完成计算任务。但是由于受到图形卡的最大纹理解析度的限制，提取的等值面可能存在精度不够的问题。

图 1 给出了相应的原理图。



(a)正交扫描方法　　(b)球面扫描

图 1　扫描方法原理图

2.3 算法流程

CPU 将处理好的二维纹理传给 GPU，同时将体数据存储为三维纹理。整个算法流程如图 2 所示，首先由 CPU 对体数据进行处理，将其转化为三维纹理并存入显存中。CPU 还要负责扫描位置起点的计算，并将结果按照前一节所描述的方法转化为二维纹理，也存入显存中待 GPU 使用。其次调用 GPU 的顶点程序，由于顶点程序不做任何处理，因此只是 Pass Through 到像素程序。在像素程序中 GPU 首先读入二维纹理上的一个像素值作为一条 Ray 的起点，然后使用 Ray-Isosurface intersection 算法计算。再取下一个起点，如此直至二维纹理上的像素被遍历。在这个过程中，Ray- Isosurface intersection 是最耗时的计算。对 CPU 而言，因为 Ray 每次前进一步，就必须使用三线性插值算法计算出其新位置的灰度值。而对 GPU 而言只需要一个简单的 sample()函数就可以达到三线性插值的效果，这里的 sample()是内置的计算，由硬件完成，效率十分高，精度可以达到 IEEE 32 位浮点数的效果。

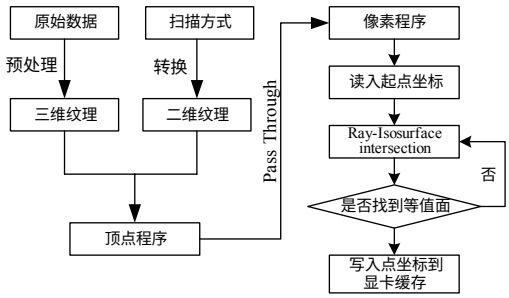


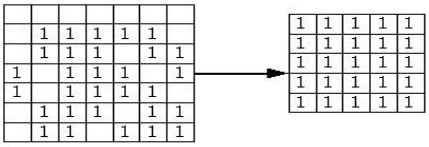
图 2　算法流程

2.4 算法的改进与优化

上面提到了使用正交投影可能带来点的重复扫描的情况。这个问题可以通过使用顺序扫描来解决。以 S_{lr}, S_{fb}, S_{ud} 分别代表左右、前后、上下 6 个扫描起始的面，那么扫描顺序可以是 S_{lr}, S_{fb}, S_{ud} 的任意排列。第 1 对扫描面可以正常进行扫描，第 2 对与第 3 对则须将先前扫描得到的结果以纹理的形式传给 GPU 程序，以 $T1,T2,\cdots,Tx$ 表示。并考察扫描得到的每个 Iso 点的法向量，如果法向量在先前完成的任何一个扫描起始面的方向上有分量，则考察这点与对应的这个方向上的 Tx 相同位置的像素值，如果二者之差小于一个常数（通常很小），就认为这点已经存在纹理 Tx 中，重复就可以避免。

在理想的情况下，算法可以正确地扫描除最外层等值面。但是在其他一些情况下，由于本文扫描方式固有的缺陷，不可能扫描到最外层结构上的每一点，因此必须用多层扫描来弥补。

多层扫描是将第 1 次扫描的结果作为新的扫描起点。这样 Ray 就可以在上次扫描到的点的位置上继续前进，直到到达终点或者认为合适为止。最后将每次扫描的结果全部绘制出来就可以得到完整的等值面。注意，随着每次扫描的进行，得到的缓存内容一般是递减的，所以，应该根据缓存内容的多少动态地计算出起始点纹理的大小。如图 3 所示，这样可以大大减少 GPU 遍历二维纹理的时间。图 3(a)为前一次的某个显卡缓存内容，将其读入内存的同时统计其有效值的个数，然后再创建新的纹理，将内存里的值再填充进去形成新的起点纹理图 3(b)。



(a)　　(b)

图 3　多层扫描

3　算法的性能分析

在最后的显示渲染过程中，使用的是四边形的点。原因在于图形卡的带宽是几何密集型的应用的瓶颈，而渲染四边形的点通常只需要传递一个顶点坐标和其法向量给图形流水线，然后图形硬件会将其光栅化为一个四边形的点。图形硬件也会自动地根据视场参数来计算这个的点的大小。如果使用其他形状的点，比如圆形的，虽然可以获得更好的显示效果，但是必须传递更多的数据给图形流水线，并且在 GPU 内使用一些算法来实现。在性能与质量两方面做出权衡之后，选择了四边形的点。

为了比较本文的新算法与传统的基于点的等值面提取算法的性能差距，笔者进行了试验。试验平台是一台普通的 PC 机，使用 Windows XP 系统，P4 2.6GHz CPU，1GB 内存，显示卡为 GF6800GT 带 256MB 的显存。试验过程中测试了提取 4 组体数据的性能参数以及显示效果。包括通过比较提取结果的点的多少来说明存储效率的不同，通过记录多次旋转物体一定角度的 FPS 值来求出平均值，最后对结果的显示效果进行了细节上的比较。表 1 给出了详细的比较结果。在表格中，同一行代表同一个数据的比较结果，在每一行中，粗体数字代表由传统的基于点的等值面提取算法的数据，而非粗体的是本算法的数据。通过观察表 1 得知，本算法提取出来的点的数量大致为传统算法的 3/5，特别是对于像头部这样封闭的结构简单的球形物体，点的数量大约可以降低到原来的 1/6。与此同时，由于需要渲染的几何图元的单位的减少，也就产生了较高的 FPS，从表的结果可以说明本算法的性能有明显提高，特别是对于 Head256 这种类似封闭球体的等值面。

表 1　本算法与传统算法的对比结果

	FPS	点数量	起始点纹理	扫描层数
Skull	52/**27**	581 713/**755 596**	400*400	3
Teapot	64/**36**	405 325/**540 789**	400*400	3
Engine	41/**34**	549 733/**576 772**	400*400	5
Head256	50/**7**	611 089/**4 248 511**	400*400	2

图 4 展示了由本文算法提取的 4 个数据的等值面。同时在图 5 中也提供了等值面提取效果的对比。

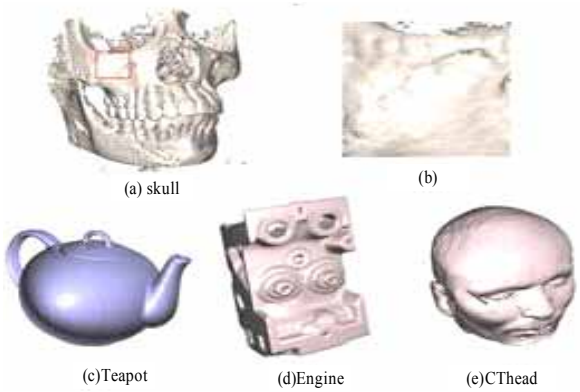


图 4　本算法提取的 4 个数据的等值面

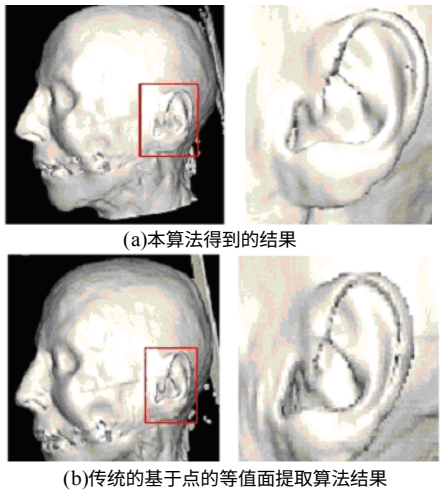


图 5　等值面提取效果的对比

可以看到，在新算法中效果有了比较明显的提高，这得益于对起始点纹理的大小控制，起始点纹理解析度越大，那么最后得到的点的结果也就越多。这就导致了密集的取样，也就是为什么视场相关的算法可以产生质量比较好的等值面的原因。最重要的，模拟三维扫描仪的原理，使用了从外而内的扫描机制，所以，最后结果虽然点的数量大大减少，但是物体可见部分较传统方法而言还是没有改变。

4　总结

本文提供了一种能够在普通 PC 机上实现的基于 GPU 加速的近似的等值面提取算法。通过使用点为基本图形单元，并且在 GPU 程序内模拟三维扫描仪的工作过程，使得等值面提取在存储效率、显示帧率和效果方面有了一定的改善。

近似提取可见等值面的是本算法的特点，这样可以减少存储开销以及在绘制时图形流水线的负担。通过模拟三维扫描过程，算法实现了一种从外而内、多层扫描的策略。通过定义起始点纹理，就完成了这个虚拟的扫描仪的扫描路线，然后在 GPU 里面模拟实现扫描仪所发出的射线对物体的扫描过程，最后将获得的结果存入显卡显存中。为了弥补单次扫描不一定可以提取到完整可见等值面的缺陷，采用了多层扫描，即以上次结果为起始，继续沿着射线方向进行扫描的方法。这样可以有效地提高算法等值面提取的精确度。文章还提到了避免计算重复点的算法，通过顺序的扫描，以及对可能重复点的检测来避免。

算法尚有不足和需要改进的地方。最关键的地方是算法对不同形状的物体的等值面提取的精度不一致。对于类似封闭球形（Head256, Teapot 等）的物体表现很好，因为这种形状的物体表面相对简单，所以往往只需两层扫描即可得到满意的结果。但是对于 Engine 这样的数据，由于整个物体结构较复杂，互相遮挡的结构较多，因此需要使用 5 次扫描才能得到比较满意的结果，所需的提取时间也有所增加。然而对于个别的数据，算法始终不能得到满意的结果。

所以，本文算法比较适用于对那些类似封闭物体，即表面结构相对简单的物体的体数据的等值面提取。

参考文献

1 Parker S, Shirley P, Livnat Y, et al. Interactive Ray Tracing for Isosurface Rendering[C]//Proc. of VIS'98. 1998.

2 Lorenson W E, Cline H E. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm[C]//Proceedings of SIGGRAPH'87. 1987, 21: 163-169.

3 Doi, Koide A. An Efficient Method of Triangulating Equi-valued Surfaces by Using Tetrahedral Cells[J]. IEICE Trans. on Commun. Elec. Inf. Syst., 1991, E-74(1): 214-224.

4 Christopher S C, Hamann B, Joy K I. Iso-splatting: A Point-based Alternative to Isosurface Visualization[C]//Proceedings of the 11th Pacific Conference on Computer Graphics and Applications. 2003.

5 NVIDIA 公司. NVIDIA 图形处理器编程指南(Version 2.2.0)[Z]. 2000.

6 杨　衡, 顾力栩. 基于图形处理器(GPU)的纹理体绘制[J]. 计算机研究与发展, 2005, 42(增刊 A): 193.