基于自适应八叉树划分的高精度四面体可视化①

童琪杰[1], 许 丽[2]

[1](浙江工业大学 计算机科学与技术学院, 杭州 310023)
[2](浙江工业大学 健行学院, 杭州 310023)

摘 要: 利用八叉树结构将四面体数据转化为规则网格数据, 能有效提高系统的交互性能. 八叉树的划分层次越高, 绘制效果越好, 但数据的存储空间以及处理时间也将大幅增多. 提出自适应的规则化表示方法来构建八叉树结构, 改进原有的单一采样策略, 并结合深度信息将采样结果转换成适用于 GPU 的八叉树纹理结构. 然后采用光线投射算法来对体数据进行绘制, 根据各区域深度不一的特点, 提出了变步长的采样绘制策略. 实验结果表明, 本文方法降低了数据的空间存储量和处理时间, 同时在绘制质量、绘制效率方面都得到了较大提高.

关键词: 四面体; 体绘制; 自适应八叉树; 光线投射; 自适应采样

High Accuracy in Visualization of Tetrahedral Volume Data Based on an Adaptive Octree

TONG Qi-Jie[1], XU Li[2]

[1](College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)
[2](Jianxing Honors College, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: Converting a tetrahedral volumetric data into regular volumetric data by an octree can improve the interactivity of the system effectively. When the depth of the octree is higher, the rendering results will be better. However, memory consumption and processing time will also increase. This paper proposes an adaptive regularization reformulation algorighm to construct the octree, improving the original single sampling strategy, combining with the depth information to transfer the sampling results into an octree texture which allows for random access in GPU. Then we use ray casting algorighm to render the regular volumetric data. Because of varying characteristics of the regional depth, the sampling algorithm responds with different step-size strategy. The experimental results show that this method reduces the memory consumption and processing time of the data, and at the same time improves the rendering quality as well as the rendering efficiency.

Key words: tetrahedron; volume rendering; adaptive octree; ray casting; adaptive sampling

1 引言

直接体绘制技术在医学、科学计算可视化等领域扮演着重要的角色, 其实质是将三维空间上的体数据经过计算、评估、转换等操作生成具有实际意义的图像. 好的体绘制技术应具有良好的交互性, 使医生或者科学家可以自由、方便地观察感兴趣的区域以及调整传输函数观察数据内部的结构信息. 随着仪器性能和科研要求的提高, 需要处理的数据量日益增多, 而现有的体绘制技术在处理大数据时效率较低, 存储空间占有量大, 不能满足实时性交互的需求, 因此, 如何结合良好的交互性手段提高绘制的效率以及减少空间占有量已成为体绘制技术的研究热点与难点.

体数据的基本单元根据其形状可分为规则数据和非规则数据. 规则体数据结构统一、分布均匀, 能较好利用 GPU 强大的并行处理能力提高绘制的效率. 非规则数据(以四面体数据为例)的分布随机性较大, 且形状各异, 大小不一, 不能有效地发挥 GPU 的并行处理能力. 主流的体绘制方法有两大类: 光线投射法和单元投影法. 它们在规则体数据可视化方面有广泛应用, 但是在绘制非规则数据时都具有某些不足. 如利用单

①收稿时间:2013-12-22;收到修改稿时间:2014-01-17

元投影算法绘制时, 每改变一次视点, 就需要对整个体数据进行重新排序, 速度较慢; 而利用光线投射算法时, 需要逐光线遍历四面体数据, 效率较低.

Leven 等[1]提出利用八叉树将四面体数据转化成半规则数据, 并存储到一张三维纹理中进行绘制, 整体上提高了绘制效率. 叶樑等人[2]在文献[1]的基础进一步提出了四面体数据的双规则化表示算法, 在绘制质量上实现了较大提高. 但该方法需要预先设置 “初级最大深度”和“高级最大深度”值, 这使其不能较好地自适应于不同类型的数据绘制; 同时, 在八叉树叶节点的采样阶段, 该方法只对采样点所在的四面体区域进行采样, 没有考虑到叶节点内其他四面体信息.

针对上述问题, 本文在叶樑[2]等人的基础上提出了基于自适应八叉树划分的高精度四面体绘制方法. 首先根据体数据的实际数量等级, 实现相应深度的八叉树统一初级划分; 然后根据叶节点内的数据特征, 自适应划分各叶节点; 最后综合利用数据重采样、GPU 八叉树纹理、光线投射、自适应采样步长等算法完成绘制. 本文方法充分利用了八叉树的自适应精细划分, 能够有效提高存储效率, 减少构建时间. 同时, 利用光线投射时的自适应采样步长可降低绘制时间获得高质量绘制结果.

文章的组织结构如下: 第 2 节讨论相关工作, 第 3 节介绍自适应规则化方法, 第 4 节描述相应的绘制方法, 第 5 节展示绘制结果, 第 6 节总结本文工作.

2 相关工作

在非规则体绘制领域, 已有不少学者提出了一些创新性方法. 如光线投射算法, 它可以追溯到 60 年代, 最初在 80 年代由 Blinn 等人[3]提出. 在 Blinn 的概念里面, 光线从屏幕的每个像素上投射出去, 当它穿入体时, 不断地进行颜色的累积. Weiler 等人[4]提出了基于硬件的光线投射算法, 它将网格信息以及网格与网格之间的连接信息都存储在纹理中, 所有计算都是在 GPU 上完成. Bernardon 等人[5]提出使用二维纹理来代替 HARC[4]中的三维纹理, 使得数据存储更加紧凑, 搜索速度更快; 接着使用基于 Tile 的方式来划分屏幕, 从而只需处理需要计算的区域; 并改进原来 HARC 中以虚拟单元填充凹处的方法, 文章提出 Depth-Peeling 方法来判断是否光线会再次进入四面体网格. Pina 等人[6]提出 ME-RAYCAST(Memory Efficient Ray Casting) 和 EME-RAYCAST(Enhanced Memory Efficient Ray Casting)算法以降低内存的开销. 由于非规则四面体本身的特性, 使得很多并行算法在绘制它们时会遇到负载不平衡, Lambronici 等人[7]以基于 Tile 的方式来达到负载平衡. 然而光线投射算法由于自身特点, 在绘制前需要将所有数据及相互邻接关系放到内存中; 在逐光线遍历四面体时, 需要进行光线与面的求交计算, 处理效率较低.

另一个主流技术是单元投影技术, 它把三维的单元投影成屏幕空间上的几何单元. 与光线投射相比, 单元投影的主要优势在于当单元进行投影时, 所有光线与单元的交点都隐含地被计算了. Shirley 和 Tuchman[8]最早提出针对非规则体数据的单元投影算法. 因为该算法只针对四面体单元, 因而它叫做 Projected Tetrahedra(PT)[9-12]. 简单地说, PT 算法将四面体投影到屏幕空间, 每个四面体被分解成多个三角形, 接着根据这些三角形得到入点标量值, 出点标量值以及长度, 查表得到颜色和不透明度, 最后按可视顺序以从后往前混合的方式累积投影到像素点上的各个颜色和不透明度. 但单元投影算法的一个瓶颈就是在绘制前需要根据可视顺序排序各个单元. 每当视点位置产生变化时, 就需要重新计算.

上述两种技术并未能达到较好的交互性. 为此, Leven 等[1]提出将四面体数据转换为八叉树层级结构, 经过采样后每个叶节点对应一个规则网格数据, 但绘制过程需要在 CPU 和 GPU 之间进行大量数据交互操作. Lefebvre 等[13]提出并实现了适用于 GPU 的八叉树纹理, 该方法用数组索引代替 CPU 中的指针, 一次性将八叉树的层级结构数据放入显存, 降低内存与显存之间数据传送的开销. 叶樑等人[2]提出四面体数据的双规则化表示算法, 首先人为定义两个不同的八叉树剖分深度, 接着用重心坐标插值方法完成数据重采样, 然后用完全空间哈希结构存储两次剖分结果之间的残差, 以实现 “数据空间”与 “数据精度”之间的平衡. 但该方法的两个不同剖分深度是人为给定的, 具有一定的局限性. 数据重采样也只是简单考虑单个四面体数据.

为克服以上不足, 本文提出自适应八叉树规则化算法以求进一步优化存储效率, 加快构建效率, 提高绘制质量.

3　基于自适应八叉树的规则化策略

八叉树的划分深度直接影响规则化的精度、存储空间和构建时间. 划分的层次越高, 采样后的结果越能逼近原始四面体信息, 但存储空间与构建时间会成倍增加. 简单的统一初次划分, 会对四面体数量或数据相关性已满足要求的部分叶节点进行不必要的分割, 造成数据的冗余, 且降低处理效率.

本文将原始数据划分成多个子集, 每个子集对应一个划分深度, 在保证规则化准确度的同时, 可以减少存储空间和时间; 在采样时, 综合考虑叶节点内所有四面体顶点信息; 然后基于文献[13]提出的利用 8 位的 RGBA 三维纹理的方法来存储八叉树结构, 保证结果的准确性.

3.1 自适应八叉树的构建策略

自适应八叉树构建分为两个阶段: 统一初级划分和逐层精细划分. 统一初级划分阶段先根据体数据的四面体个数确定初级八叉树划分深度, 然后根据该深度值对体数据进行统一划分. 初级统一划分之后, 各叶节点内的区域信息量可能出现较大差异. 此时, 根据叶节点内的四面体数以及它们顶点的属性值的方差来判断是否需要进行逐层精细划分. 对包含信息量较小、各四面体相关性较高的叶节点可直接采样; 而仍存在较大信息量的叶节点, 则需要进行逐层精细划分, 直到该节点都满足数量、相关性方面的要求或者当前叶节点达到指定的最大划分深度. 自适应八叉树的构建过程如图 1 所示(为了方便起见, 我们以二维平面上的四叉树为例). 首先用根节点 1 包含所有四面体数据, 接着通过计算以深度 1 作为统一初级划分深度进行划分, 得到节点 2,3,4,5. 此时进入逐层精细划分阶段, 通过计算节点 2,3,4,5 内的四面体数以及属性值的方差, 得知节点 2,4 需要继续划分, 节点 3,5 直接采样; 之后, 再次计算新生成叶节点内的四面体数和属性值的方差, 对节点 7 继续划分, 其他叶节点直接采样. 图 2 为自适应划分后相应节点所对应的二维子区域示意图.

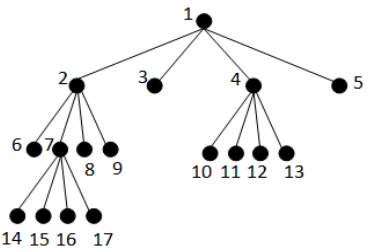

图 1　自适应划分的二维示意图

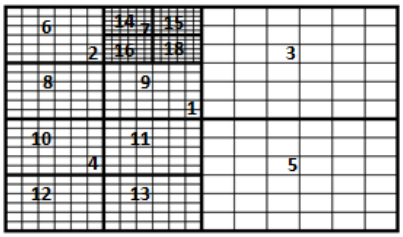

图 2　自适应划分后体数据各子区域的二维分布

3.2 叶节点处的采样策略

对四面体数据进行自适应八叉树构建之后, 需要对各叶节点内的数据进行采样. 每个叶节点都包含有一定数量的四面体. 文献[2]中的方法只考虑采样点所在四面体的标量信息, 没有考虑该叶节点内的其他四面体标量信息对最终结果的影响. 本文基于体绘制的辐射-吸收模型[14], 从整体上进行考虑, 提出了新的采样策略. 体绘制的辐射-吸收模型, 如图 3 所示, 一定数量的辐射能量在 t=d 处辐射, 并且沿着路径 d 被部分吸收.

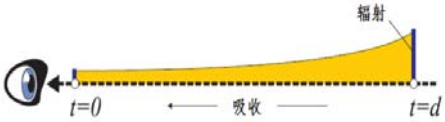

图 3　体绘制的辐射-吸收模型

文献[15]分析了多种计算梯度方法的优劣性, 总结得出 Inverse Centroid Distance 算法计算复杂度低, 效果好. 辐射-吸收模型与该算法思想相符, 所以本文借鉴该算法思想提出了如下的采样策略: 1). 先计算采样点与节点内其它顶点的距离; 2). 以距离的倒数为系数进行加权累积. 具体计算公式如下:

$$S(x_0)=k\times\sum_{i=1}^{n}\left(\frac{1}{D(x_i)}\times S(x_i)\right) \qquad (1)$$

其中, $S(x_0)$为最终的采样值, x_0 为采样点坐标, x_i 为第 i 个顶点的空间坐标, $S(x_i)$为第 i 个顶点的标量值, $D(x_i)$为第 i 个顶点到采样点的距离, n 为当前叶节点内的顶点数, k 为常数.

3.3 采样值的存储策略

所有叶节点采样结束之后, 需要对采样结果进行存储. 文献[1,2]提出将采样结果转换为适用于 GPU 的纹理结构. 纹理的元素为四元组(对应于 RGBA 通道), 其中 RGB 三个分量存储第一个子节点的索引值或者叶节点体数据的采样值, A 分量存储当前节点的类型.

节点的类型分为空节点、中间节点和叶节点. 若该节点为空节点, 则 A=0; 若为中间节点, 则 A=255; 若为叶节点, 则 A=128. 该方法没有考虑叶节点的深度信息, 不适用于后续绘制时光线投射的自适应变步长算法. 因此, 本文在该方法基础上进行了适当改进. 若节点为空节点或者中间节点, 则存储方式不变. 若节点为叶节点, 则用 A 来记录深度信息. 因为深度值 d 一般小于等于 10, 直接将 A 设为 d 会在归一化时产生不必要的误差, 所以, 本文将 A 设为 10d.

3.4 自适应规则化流程

八叉树辅助结构能快速定位采样点的相关四面体, 并使采样结果精确逼近原始四面体数据, 但在自适应规则化时必须考虑存储效率和采样精度这两个因素, 具体算法如下:

1) 假设四面体数据在空间上是均匀分布的, 预计算统一初级划分深度 d. 给定阀值 U, 当某一深度 d 下叶节点内的平均四面体数小于 U, 则将深度 d 作为统一初级划分的深度. 利用文献[1]的方法构建深度为 d 的八叉树.

2) 对八叉树初级结构进行逐层精细划分, 生成自适应八叉树. 逐个计算叶节点内的四面体数量, 如果四面体数量小于给定阀值 V, 则不需要继续划分; 否则, 进一步计算叶节点内所有四面体顶点的标量值的方差 c, 如果 c 小于给定阀值 W, 说明这些四面体的属性已较为相似, 不需要继续划分. 其余情况下, 沿 xyz 轴方向二分当前叶节点. 对于新生成的所有叶节点, 重复上述处理过程, 直到所有叶节点不需要继续划分, 或者达到指定的最大划分深度.

3) 利用公式(1)对八叉树的各叶节点进行采样.

4) 根据 3.3 所述的采样存储策略将采样结果转换成适用于 GPU 的纹理结构.

4 绘制

以光线投射方法绘制规则体数据能获得高质量的绘制结果. 采用自适应规则化之后, 原始四面体数据已转化为半规则数据, 并存储在八叉树的纹理结构中. 因此, 本文采用光线投射算法来对体数据进行绘制, 且根据各区域深度不一的特点, 提出了变步长的采样绘制策略.

4.1 自适应采样步长

经典的光线投射算法是从视点出发向屏幕上每个像素点投射一条光线, 沿着光线方向以固定步长进行采样, 采样值通过传输函数映射为相应的颜色和不透明度, 随后相对视点位置按公式(2)、公式(3)[15]以从前往后的顺序进行颜色和不透明度的混合, 得到像素值.

$$C'_i = C'_{i-1} + (1 - A'_{i-1}) \times C_i \quad (2)$$

$$A'_i = A'_{i-1} + (1 - A'_{i-1}) \times A_i \quad (3)$$

经过自适应八叉树规则化处理后, 原始数据空间被划分成多个大小不一的子空间. 细节较多的区域由更多、更小的叶节点表示. 以固定步长采样数据, 有可能会遗漏部分重要细节, 也可能会增加不必要的计算开销. 如图 4 所示, 以 L_1 或者 L_2 为采样步长, 会遗漏区域 C 的信息; 以 L_3 为采样步长时, 则会增加区域 A, B, D 的计算开销. 其中, $L_1=2\times L_2$, $L_2=2\times L_3$.

本文根据采样点所属节点在八叉树上的深度, 自动调整采样步长. 假定视点的深度 d_s=1, 给定初始采样步长为 L_s, 当某一光线进入区域 Z 时, 获得当前采样点 P. 根据 P 的空间位置查找八叉树的三维纹理表得到其所属叶节点的 A 分量值, 即 P 的深度 d. 然后判断 d 是否等于前一采样点的深度值, 若相等, 则采样步长不变, 继续采样; 若不等, 则回到前一采样点, 利用公式(4)计算新的采样步长, 并得到新的采样点位置. 反复迭代上述过程, 直到新得到的采样点的深度小于或者等于当前采样点深度.

$$L = L_s / 2^{d-1} \quad (4)$$

其中, L 为实际采样步长, L_s 为初始给定步长, d 为叶节点的深度值.

以图 4 为例, 采用上述方法进行采样时, 光线 R 的最终采样点分别为 P_1,P_2,P_3,P_4,P_6. 相比于固定步长采样方法, 本文方法不仅保证了结果的正确性, 还一定程度提高了绘制效率.

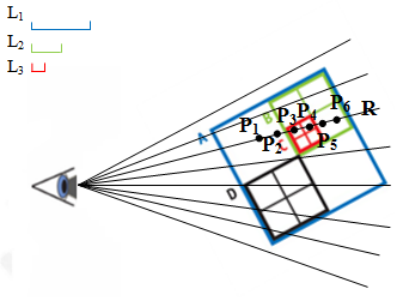

图 4 最大深度为 3 的光线投射示意图

4.2 颜色累积方法

直接体绘制技术利用传输函数将体数据上的标量值映射为相应的颜色和不透明度，一般的预积分方法将标量值对应的传输函数值离散近似成一张二维表，以查表的方式得到结果再进行颜色累积. 当采样步长不固定时，需要建立多张二维表来对应每个步长，且每改变一次传输函数就要重新计算这些表. 计算量大，交互性差. 针对这些缺点，Moreland 等人[16]提出了部分预积分算法，该算法利用分步积分法对原始的体数据累积光强积分公式进行变形，使部分预积分的查找表不依赖于传输函数，且采用步长 D 的取值可以任意. 本文采用该方法来进行颜色和不透明度的累积.

5 实验结果

实验配置如下：CPU:Intel® Core™ i5-2310 2.90GHz，内存：4G，图形卡：NVIDIA GeForce GTX 550Ti. 本文采用的 CUDA 版本为 4.1，绘制图像分辨率均为 512*512.

本文选用 Blunt 和 Spx2 这两个数据进行实验测试. 它们的具体参数如表 1 所示.

表 1　测试数据集参数

数据集	四面体个数	顶点数
Blunt	187k	41k
Spx2	828k	149k

5.1 自适应划分的深度对构建时间和存储空间的影响

本文对两个测试数据分别以最大划分深度 8 层，9 层，10 层进行自适应八叉树构建，如表 2 所示，当最大划分深度增大时，虽然能较好地精细部分区域，但也会过多反复细分部分细小区域，增加八叉树构建时间和存储空间，为了平衡“处理效率”和“数据精度”，本文设定最大划分深度 D_{max}=9. 若利用文献[2]中的方法以 8 层作为初级最大深度，对两个测试数据进行八叉树构建，构建时间以及最终的节点数如表 3 所示. 通过比较表 2 和表 3 可知，当划分深度都为 8 层时，本文的方法能进一步优化处理效率和空间利用率；当利用本文方法将最大划分深度设为 9 层时，虽然构建时间和存储空间与文献[2]中的方法差不多，但是我们的数据能更精确地表示重要区域信息，数据的质量更高.

表 2 采样本文自适应八叉树规则化四面体数据

最大划分深度 / 构建时间以及最终节点数 / 数据集	8 层	9 层	10 层
Blunt	24.13s, 138k	35.34s, 208k	43.77s, 251k
Spx2	81.64s, 247k	122.65s, 376k	154.12s, 452k

表 3　采样文献[2]中的方法规则化四面体数据

	构建时间	最终节点数
Blunt	36.69s(初级最大深度为 8 层)	214k
Spx2	127.34s(初级最大深度为 8 层)	367k

5.2 最终效果的比较

接着，采用光线投射算法和自适应变步长算法对规则化后的数据进行绘制，图 6, 8 分别为 Blunt 数据和 Spx2 数据的绘制效果，其中最大划分深度为 9 层. 图 5，7 分别为改进前测试数据所对应的绘制效果，初级最大深度为 8 层. 通过比较，可以看出本文提出的基于自适应八叉树规则化的四面体可视化算法能获得较好的绘制质量.

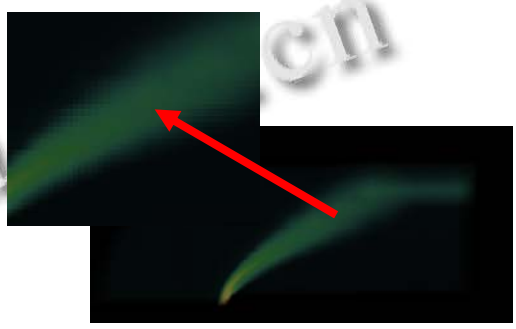

图 5　初级最大深度为 8 层的 Blunt 数据绘制效果

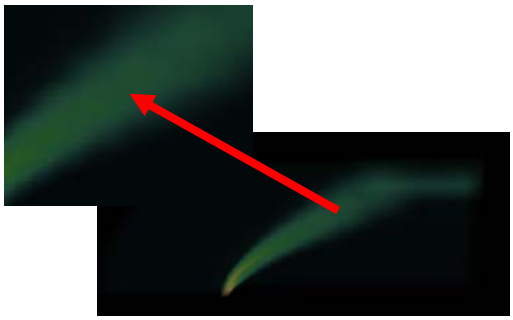

图 6　最大划分深度为 9 层的 Blunt 数据绘制效果

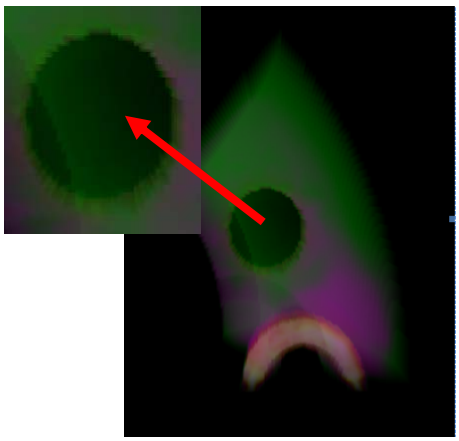

图 7 初级最大深度为 8 层的 Spx2 数据绘制效果

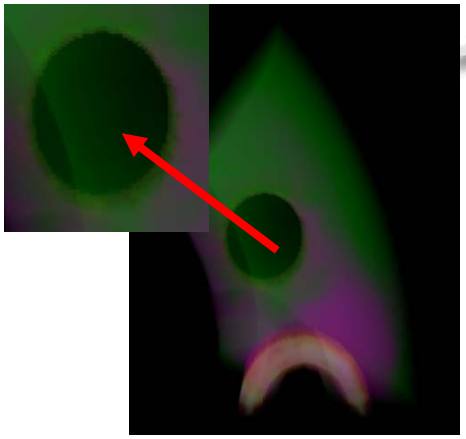

图 8 最大划分深度为 9 层的 Spx2 数据绘制效果

6 总结

本文提出了基于自适应八叉树的规则化方法, 提高了规则化体绘制的绘制质量. 该方法的核心在于, 通过自适应划分使每一个叶节点对应一个划分深度, 在保证规则化准确度的同时, 可以减少存储空间. 实验结果显示, 该方法能有效提高四面体数据的绘制质量.

参考文献

1 Leven J, Corso J, Cohen J, Kumar S. Interactive visualization of unstructured grids using hierarchical 3D textures. In: Johnson C, ed. Proc. of the 2002 IEEE Symposium on Volume Visualization and Graphics. New Jersey. IEEE Press. 2002. 37–44.

2 叶棵,李昕,王桂珍,陈海东,陈为,周敏.适用于 GPU 的四面体体数据规则化与可视化.计算机辅助设计与图形学学报,2011,23(6):933–940.

3 Blinn JF. Light reflection functions for simulation of clouds and dusty surfaces. In: Bergeron RD, ed. SIGGRAPH'82, Proc. of the 9th Annual Conference on Computer Graphics and Interactive Techniques. New York. ACM Press. 1982. 21–29.

4 Weiler M, Kraus M, Merz M, Ertl T. Hardware-based ray casting for tetrahedral meshes. In: Bartz D, ed. Proc. of IEEE Visualization 2003. Seattle, WA. IEEE Press. 2003. 333–340.

5 Bernardon FF, Pagot CA, Comba JLD, Silva CT. GPU-based tiled ray casting using depth peeling. Journal of Graphics, GPU, and Game Tools, 2006, 11(4): 1–16.

6 de Pina AA, Bentes C, Farias R. Memory efficient and robust software implementation of the raycast algorithm. In: Rossignac J, ed. The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2007. Plzen, Czech Republic. Science Press. 2007.

7 Labronici BB, Bentes C, Drummond LMA, Farias R. Dynamic screen division for load balancing the raycasting of irregular data. Proc. of the IEEE Cluster Computing and Workshops. New Orleans, LA. IEEE Press. 2009. 1–10.

8 Shirley P, Tuchman A. A polygonal approximation to direct scalar volume rendering. In: England N, ed. VVS'90 Proc. of the 1990 Workshop on Volume Visualization. New York. ACM Press. 1990. 63–70.

9 Wylie B, Moreland K, Fisk LA, Crossno P. Tetrahedral projection using vertex shaders. In: Johnson C, ed. VVS'02 Proc. of the 2002 IEEE Symposium on Volume Visualization and Graphics. Piscataway, NJ. IEEE Press. 2002. 7–12.

10 Weiler M, Kraus M, Merz M, Ertl T. Hardware-based view-independent cell projection. IEEE Trans. on Visualization and Computer Graphics, 2003, 9(2): 163–175.

11 Kraus M, Qiao W, Ebert DS. Projecting tetrahedra without rendering artifacts. In: Alexa M, ed. VIS'04, Proc. of the Conference on Visualization'04. Washington. IEEE Press. 2004. 27–34.

12 Marroquim R, Maximo A, Farias R. GPU-based cell projection for interactive volume rendering. Brazilian Symposium on Computer Graphics and Image Processing SIBGRAPI'06. Manaus. IEEE Press. 2006. 147–154.

13 Pharr M, Fernando R. GPU Gems II. 2nd ed. Boston: Addison-Wesley, 2005: 595–613.

14 Hadwiger M, Ljung P, Salama CR, Ropinski T. Advanced illumination techniques for GPU volume raycasting. In: Hadwiger M, ed. ACM SIGGRAPH ASIA 2008 Courses. New York. ACM Press. 2008. 1.

15 Correa CD, Hero R, Ma KL. A comparison of gradient estimation methods for volume rendering on unstructured meshes. IEEE Trans. on Visualization and Computer Graphics, 2011, 17(3): 305–319.

16 Moreland K, Angel E. A fast high accuracy volume renderer for unstructured data. VVS '04, Proc. of the 2004 IEEE Symposium on Volume Visualization and Graphics. Washington. IEEE Press. 2004. 13–22.