基于光线投射算法的混合场景可视化

袁非牛[1] 廖光煊[1] 范维澄[1] 周荷琴[2]

[1](中国科学技术大学火灾科学国家重点实验室,合肥 230027) [2](中国科学技术大学信息科学与技术学院,合肥 230027)

摘 要 体绘制技术常用于 3 维体数据场的可视化,其虽然可以生成高质量的投影图像,但通常不能绘制由体数据与点、线、面图形组成的混合场景。在现有的混合场景可视化方法中,有些只能绘制由体数据与面图形组成的复杂混合场景,而不能处理存在点和线的混合场景;有的则成像速度慢、成像质量差。为了能够正确地绘制复杂混合场景,采用 SIMD 和软件加速等技术,提出了一种速度快、成像质量高的基于光线投射算法的混合场景可视化方法,并分析了该算法所具有的 3 种绘制次序,以便满足不同应用的要求。该算法既可用于不同场景的绘制,又可用于平行和透视投影中。实验结果表明,该算法能够正确地绘制体数据与点、线、面图形组成的混合场景,且成像速度快,图像质量高。

关键词 3 维可视化 体绘制 光线投射 图形图像处理

中图法分类号:TP391.4 **文献标识码**: A **文章编号**: 1006-8961(2005)07-0850-06

Visualization of Hybrid Scenes Based on Ray Casting Algorithm

YUAN Fei-niu[1], LIAO Guang-xuan[1], FAN Wei-cheng[1], ZHOU He-qin[2]

[1](*State Key Lab of Fire Science, University of Science & Technology of China, Hefei* 230027)

[2](*School of Information Science & Technology, University of Science & Technology of China, Hefei* 230027)

Abstract Volume rendering is often used for 3D volumetric datasets visualization. It can generate high quality images, but can not render volumes and polygons combined scenes. Among many currently available techniques of visualizing hybrid scenes, some can render volumes and triangles combined scenes, but can not process geometric points and lines. To correctly render these complicated hybrid scenes, a ray casting based algorithm using SIMD and software acceleration techniques is presented. The algorithm has three types of rendering order, each of which has its own advantages and drawbacks for different applications. It can be adopted in parallel and perspective projection for many applications. Experiments show that the algorithm can properly visualize volumes, polygons, lines and points combined scenes, and rendered images have high quality and its rendering speed is very fast.

Keywords 3D visualization, volume rendering, ray casting, computer graphics & image processing

1 引 言

随着断层扫描、数值计算等技术的发展,产生了各种各样的 3 维离散数据场。这些数据大致分为以下 3 类:第 1 类是科学计算生成的 3 维数据场,如流体计算、有限元分析、数值模拟等方式产生的体数据;第 2 类是通过各种断层扫描设备获取的数据,如通过 CT、MRI 等设备扫描而得到的体数据;第 3 类是采用计算机对传统多边形几何实体进行体素化(voxelization)得到的体数据。大家知道,从不同的角度,体数据场有不同的分类,例如,根据数据场中采样点的空间几何分布特征,数据场可以分为规则数据场(regular)和非规则数据场(irregular)等[1]。

基金项目:国家自然科学基金专项基金项目(50323005);中国博士后科学基金项目 (2004036155)

收稿日期:2004-06-15;**改回日期**:2005-05-08

第一作者简介:袁非牛(1976 ~),男。1998 年和 2001 年分别获得合肥工业大学工学学士和硕士学位,2004 年获中国科学技术大学工学博士学位,现为中国科学技术大学火灾科学国家重点实验室博士后。主要研究方向为基于计算机视觉的火灾探测、3 维可视化、医学影像图像处理。已发表论文 10 余篇。E-mail: yfn@ ustc. edu. cn

鉴于断层扫描设备等获取的 3 维体数据是规则数据场,且应用非常广泛,因此本文主要讨论这种规则、标量的 3 维数据场与点、线、面图形组成的混合场景可视化问题。

3 维标量体数据的可视化方法主要有面绘制和体绘制[2~4]两大类,其中面绘制在提取等值面片的过程中,由于丢失了体数据的大量细节信息,因而绘制的图像质量不高,而且面片数量一般比较巨大,有时会超出图形硬件的处理能力。此外,当阈值改变时,还必须重新进行耗时的等值面片提取过程,不宜交互调整阈值。而体绘制则由于不必把等值面转化为中间几何图元的表示形式,而是根据特定的转换函数直接进行绘制,就可以表现体数据中的细腻结构,所以体绘制更适合用于对图像质量要求较高的场合,其缺点是成像速度慢。

但在实际应用中,并不只是单纯地存在体数据,往往是体数据与传统面图形的混合,例如医学应用中的多边形切割、手术仿真等。要解决混合场景的成像问题,一种比较容易想到的方法,就是先把几何图形转换成体数据的统一格式,然后再采用常规体绘制算法进行绘制。这种方法首先把多边形网格拟合到体数据场,然后通过 3 维扫描转换[5]来生成标量体数据。由于这种方式会导致几何对象精度大大降低,因而容易产生锯齿等现象。Levoy 提出了一种混合光线跟踪算法[6],能够解决多边形与体数据混合场景的可视化。该方法在光线跟踪时,需要求出光线与多边形的交点,然后才能将交点处的颜色、不透明度与体数据采样得到的颜色、不透明度进行合成。但是,对于数学上没有宽度的点和线,由于大多数情况下投射的光线与点、线几何对象不相交,也无法计算点和线对结果图像的贡献,所以该算法不适合处理存在点和线的混合场景。Kreeger 等采用硬件 3 维纹理贴图与 Z-Buffer 实现了混合场景的绘制[7],虽然可以进行快速的渲染,但由于采用了 Pre-Shaded 技术,从而导致结果图像质量不高。Schmidt 等研究了基于 Shear-warp 算法的混合场景成像问题[8],但因采用了前后两组深度缓存,从而增加了额外开销。

为了解决上述问题,本文提出了一种基于光线投射的混合场景可视化算法,它可以绘制点、线、面图形与体数据组成的混合场景,文中还分析了这种算法 3 种可能的绘制次序及其优缺点。该算法由于采用了奔腾 SIMD、分割[9]和 OpenGL 等技术,因而成像速度比较快,基本上可以满足实际应用的要求。

2　方　法

光线投射算法(ray casting)是一种图像空间的绘制算法,其虽成像质量高,但速度慢。为了获得高质量图像,光线投射算法仍然被广泛地应用在各种场合。因已有多种加速技术[4,9,10]用于算法的加速,这就极大地提高了绘制速度。鉴于传统光线投射算法只能绘制体图形,却无法处理面图形,为此,本文在光线投射算法基础上增加了计算深度缓存、OpenGL 绘制等模块,以使得它可以处理混合场景。在混合场景中,由于图形对象只有体图形(体数据)和面图形(点、线、面)两类,因此,根据体图形和面图形的绘制次序,将本文算法分为以下 3 种情况进行讨论:

(1)先体后面,即先采用光线投射算法绘制体图形,然后采用 OpenGL 绘制面图形;

(2)先面后体,即先采用 OpenGL 绘制面图形,然后采用光线投射算法绘制体图形;

(3)不分先后,即体图形和面图形同时绘制,不分先后。

在本文的算法中,由于体图形和面图形是通过深度缓存信息进行正确消隐与融合的,因此,先绘制的对象不能为半透明,否则无法得到正确的深度信息,而后绘制的对象可以为半透明。对于不分先后次序的绘制方式,体数据和面对象均可以为半透明,但不能处理同时存在点或线的场景。

2.1　先体后面

这种绘制次序先渲染体图形,然后再绘制面图形,所以只能绘制不透明的体图形(绘制时,采用二值转换函数),但可以绘制不透明和半透明的面图形。为了在两种截然不同的算法上进行正确地消隐,必须在绘制体图形对象时,保存屏幕深度(即 Z-Buffer)。传统图形绘制算法采用了硬件比较容易实现的 Z-Buffer 消隐算法,该算法是根据屏幕深度的大小来确定遮挡关系的。设近剪切面距离为 z_n,远剪切面距离为 z_f,实际场景深度值为 z_e(即眼睛(eye,即 z_e 下角 e)坐标系中的坐标 z),则 Z-Buffer 算法中的屏幕深度 z_S(即屏幕(screen,下同)坐标系中的坐标 z)为

$$z = \frac{z_f}{z_f - z_n} \cdot \left(1 - \frac{z_n}{z_e}\right) \tag{1}$$

光线投射算法中的深度 z_r(下角 r 即 raycasting)与 Z-Buffer 缓存中的深度 z 的几何意义完全不同(如

图 1 所示)。光线投射算法中的深度是指光线从眼睛到不透明等值面之间的实际距离,而 Z-Buffer 缓存中的深度则是根据式(1)计算而得到的数值,并且已被归一化到[0,1]。

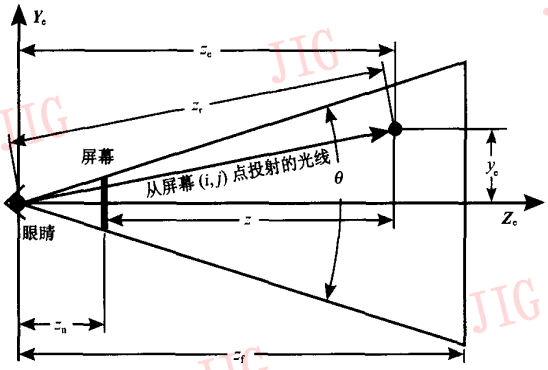


图 1 Z-Buffer 深度 z 与光线投射深度 z_r 关系

(摄像机 X 轴未绘出)

Fig. 1 Comparison between Z-Buffer depth and ray casting depth

在图 1 中,成像平面位于近剪切面位置处,屏幕中心点($W_S/2$,$H_S/2$)位于摄像机 Z 轴上。根据式(1)和几何关系,Z-Buffer 中的深度 z 与光线投射算法中的深度 z_r 之间的关系为

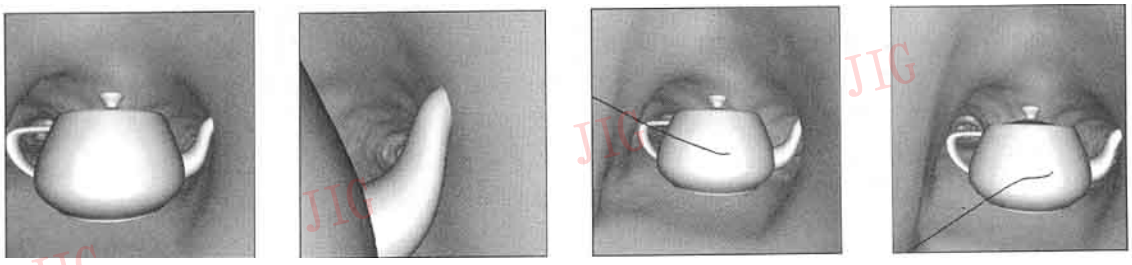
$$z=\frac{z_f}{z_f-z_n}\cdot\left(1-\frac{z_n}{z_r}\cdot\sqrt{1+\frac{\tan^2(\theta/2)}{H_S^2}\cdot[(2\cdot i-W_S)^2+(2\cdot j-H_S)^2]}\right) \tag{2}$$

其中,z_n、z_f 为近、远剪切面距离,z_r 为光线投射算法得到的深度,z 为归一化后的 Z-Buffer 深度,θ 为摄像机 y 向视角,W_S、H_S 分别为屏幕的宽度和高度,i、j 分别为图像像素水平和垂直坐标,即索引号。

因为在后续的面绘制阶段需要按照深度值进行消隐,所以在光线投射算法中必须计算 Z-Buffer 深度值。根据式(2)就可以计算出与给定的光线投射深度 z_r 对应的 Z-Buffer 深度 z。完成了体数据对象的绘制后,即可得到体数据对象的深度缓存,而且在随后的多边形面绘制阶段还需要利用这个深度信息进行正确的消隐与融合。大家知道,多边形的绘制技术已经非常成熟,已经有了许多非常优秀的图形渲染引擎,如 OpenGL、Direct 3D 等。在本文算法中就采用 OpenGL 技术,即首先,将光线投射算法得到的深度缓存写入 OpenGL 的 Z-Buffer 缓存中,然后再绘制点、线、面图形对象,算法伪代码如下:

```
//绘制体图形
For y: = 1 to 帧缓存高度 do
Begin
  For x: = 1 to 帧缓存宽度 do
  Begin
  从成像平面(x,y)处投射一条光线,搜索等值面;
  找到等值面后,计算帧缓存(x,y)处色彩值;
  根据式(2)计算深度缓存(x,y)处深度值;
  End
End
//绘制面图形
将体绘制阶段得到的帧缓存、深度缓存分别写到显卡硬件中的帧缓存和深度缓存;
启用光照、深度测试等;
绘制点、线、面场景
```

图 2 是采用“先体后面”方式绘制的图像。场景中含有一个 512×512×112 的人体 CT 体数据,还包含一个三角形网格茶壶和一条 3 维曲线图形。从图中可以看出,消隐效果非常好,图像质量比较高。



(a) 气管体数据与茶壶 (b) 气管体数据与茶壶 (c) 气管体数据和茶壶、曲线 (d) 气管体数据和茶壶、曲线

图 2 “先体后面”方式绘制的图像

Fig. 2 Rendered images by rendering volume prior to polygons

2.2 先面后体

在这种方式中,先用 OpenGL 绘制面图形,然后采用光线投射算法绘制体数据,两者之间的联系仍然是深度缓存。由于是先绘制多边形对象,因此需要将

Z-Buffer 中的深度 z 变换到光线投射算法中的深度 z_r,根据式(2)就可求出 z_r 与 z 的映射关系(式(3))。

OpenGL 绘制完面图形对象后就生成了深度缓存 Z-Buffer,然后应用式(3)将 Z-Buffer 的深度变换为与光线投射算法对应的深度。如图 3 所示,图中的虚线表示因多边形绘制而产生的深度缓存。在随后的体数据绘制过程中,每投射一条光线,只需跟踪到深度缓存位置(图 3 的虚线位置)即可,这样就可以通过提前终止光线的传播来减少重采样次数和缩短成像时间。例如,图 3 中的光线 1 只需跟踪到深度缓存边界处,就提前终止了光线的传播。由此可见,"先面后体"绘制方式成像速度比较快,甚至比单纯的体数据场景成像速度还要快。

$$z_r = \frac{z_f \cdot z_n}{z_f - (z_f - z_n) \cdot z} \cdot \sqrt{1 + \frac{\tan^2(\theta/2)}{H_S^2} \cdot [(2 \cdot i - W_S)^2 + (2 \cdot j - H_S)^2]} \tag{3}$$

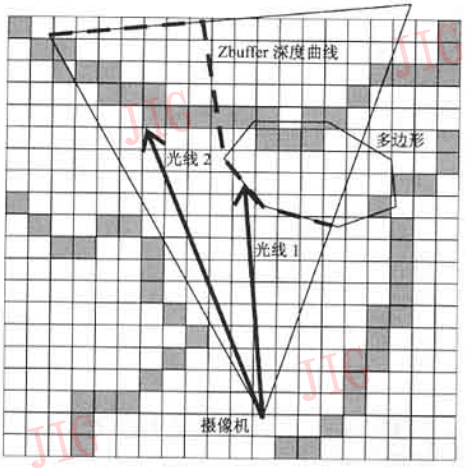


图 3　"先面后体"方式中的光线投射示意图

Fig. 3　First rendering volume prior to polygons

2.3　不分先后

这种方式和 Levoy 提出的算法[6]有点类似,即需要求出屏幕上每一点投射的光线与多边形的交点,然后将交点按深度进行排序,并把光线与多边形交点处的着色颜色 C 与不透明度 α 看成是体数据内一个采样点的贡献(图 4),这样就能够正确地进行消隐。

从图 4 中可以看出,由于交点的引入破坏了原始采样点的等间隔分布属性,因此在合成时需要进行额外的处理,如进行插值、过采样等[6]。由于多边形边界处容易产生锯齿现象,因此为了提高图像质量,还需要进行抗锯齿等处理。由于多边形与光线的求交、着色均采用软件方式完成,且没有利用图形硬件功能,所以这种绘制次序的成像速度比较慢。虽然该方式可以处理半透明的体和面图形的混合场景,但因大多数情形下,光线和点、线对象之间没有交点,故无法计算点、线图形对象对结果图像的贡献,这样也就不能够绘制同时存在点、线的混合场景。

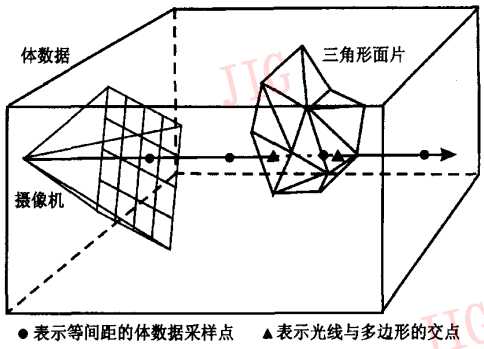


图 4　"不分先后"方式中的光线投射

Fig. 4　Rendering polygons and volume in non-prior order

综合上述讨论,现把各种绘制次序的优缺点、适用范围列于表 1 中。由于根据场景复杂程度和应用的不同,采用速度优先等策略可以自动地选择不同的绘制次序,因此对用户来讲完全透明,也简化了应用的难度。

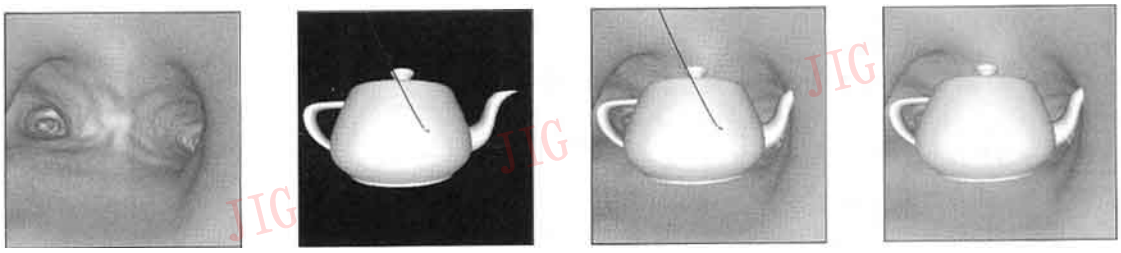
表 1　各种绘制次序的比较

Tab. 1　Comparison between the three rendering orders

	先体后面	先面后体	不分先后
体图形(体数据)	不透明	不透明、半透明	不透明、半透明
面图形(多边形)	不透明、半透明	不透明	不透明、半透明
点、线	可以	可以	不可以
绘制速度	中	快	慢

3　实验及结果

为了验证算法的效果,在 Visual C++ 开发环境下,采用奔腾 SIMD、阈值分割和 OpenGL 等技术,实现了本文提出的成像算法,并在一台 2.8GHz 的 PC 机上进行了实验。实验采用的数据是由 512 × 512 × 112 的人体气管 CT 体数据、一条气管导航中心线和一只三角形网格茶壶组成的混合场景,渲染结果图像如图 5 所示,绘制时间列于表 2 中。



(a) 气管体数据 (b) 茶壶与曲线 (c) 气管体数据和茶壶、曲线 (d) 气管体数据和茶壶

图 5 本文算法绘制的图像

Fig. 5 Rendering images with the algorithm

表 2 渲染图像大小为 512×512 时的绘制时间

Tab. 2 Rendering times with image size of 512×512

	非混合场景		混合场景（体数据和茶壶与曲线）		
	体数据	茶壶与曲线	先体后面	先面后体	不分先后
绘制时间(s)	0.219	<0.030	0.266	0.203	0.967

图 5(a)是人体 CT 气管体数据单独渲染的结果图像，本文算法绘制时间为 0.219s，图 5(b)是传统茶壶与曲线图形数据的渲染图像，OpenGL 绘制时间小于 0.030s，图 5(c)是气管体数据和茶壶、曲线组成的混合场景成像结果。对于图 5(c)场景，“先体后面”方式绘制时间为 0.266s，“先面后体”方式绘制时间为 0.203s，“不分先后”方式绘制时间为 0.967s。由于“不分先后”方式不能处理点和线图形对象，因此对图 5(c)中的混合场景成像后，曲线无法显示，其得到的结果图像如图 5(d)所示。

根据表 2 就可以分析出绘制次序对成像速度的影响。从表 2 可以看出，成像速度最快的方式是“先面后体”方式，这是因多边形的遮挡缩短了光线的传播距离，从而减少了大量不必要的重采样次数。“先体后面”绘制方式次之；“不分先后”方式成像速度最慢，其主要是因为它必须求出每条光线与多边形的交点，而且是采用软件方式进行着色，不能利用图形硬件的缘故。从结果图像上可以看出，各种绘制次序不影响成像质量，只影响成像速度。“不分先后”方式与 Levoy 的算法[6]类似，但本文算法比 Levoy 等人提出的方法速度快、成像质量高、适用范围更广。另外，虽然基于 3 维纹理帖图的技术也可以实现混合场景的绘制[7,11]，但这种类型算法需要高档图形硬件支撑才能获得快速的成像速度，且成像质量不如本文算法好。

图 6 是用“先体后面”方式来绘制曲线、半透明

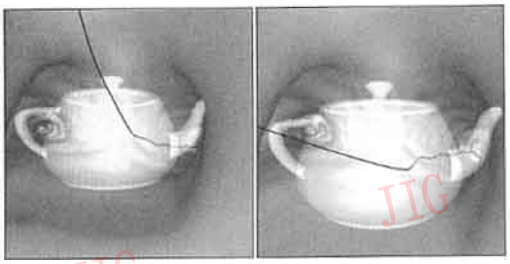


图 6 半透明绘制结果图像

Fig. 6 Translucent images by rendering volume prior to polygons

茶壶与体数据场景所生成的渲染图像。图 7 是用“不分先后”方式绘制的由美国数字人彩色体数据和切割平面组成的混合场景的结果图像。为了得到切割效果，本文还采用了一种简单的绘制技巧，即把切割平面组成的实体看成完全透明，也就是不透明度为 0，这样就能屏蔽实体内体数据的贡献。实验结果证明，本文算法不仅成像质量高，而且消隐、融合效果好。

图 7 美国数字人 VHM 和切割多边形

Fig. 7 Rendering images of Visible Man

4 结 论

3 维标量体数据采用体绘制技术虽然可以获得

高质量的投影图像,但不能够处理体数据与点、线、面图形组成的混合场景。现有的混合场景可视化方法,有的只能够很好地处理体数据与面图形的混合场景,而不能处理点和线的情况。本文提出了一种混合场景成像方法,它能够很好处理体数据与点、线、面图形组成的混合场景,文中还分析了 3 种不同的绘制次序及其应用场合。实际应用表明,该算法既可以用于平行投影,又可以用于透视投影,生成的图像质量很高。实验表明,绘制次序对成像速度和适用范围有很大的影响,"先体后面"方式只能够处理不透明体数据和不透明与半透明点、线、面图形,成像速度居中;"先面后体"方式只能够绘制不透明点、线、面图形和不透明与半透明体数据,成像速度最快;"不分先后"方式最自由,虽然其可以渲染不透明与半透明面图形和不透明与半透明体数据,但不能处理点、线图形对象,且成像速度最慢。本文算法由于是建立在笔者提出的基于奔腾 SIMD 和分割技术的光线投射算法[9]基础上,因此成像速度快,另外还采用 3 种绘制次序,不仅拓展了适用范围,而且能够满足实际应用的要求。

参考文献(References)

1 Sramek Milos, Kaufman Arie. Fast ray-tracing of rectilinear volume data using distance transforms[J]. IEEE Transactions on Visualization and Computer Graphics, 2000, **6**(3): 236 ~ 252.

2 Lorensen W, Cline H. Marching cubes: a high resolution 3D surface construction algorithm [J]. Computer Graphics, 1987, **21** (4): 163 ~ 169.

3 Lee Westover. Footprint evaluation for volume rendering[J]. ACM Computer Graphics, 1990, **24**(4): 367 ~ 376.

4 YUAN Fei-niu, ZHOU He-qin, ZHAO He, *et al.* Sampled points decomposing based ray casting for virtual endoscopy [A]. In: Proceedings of SPIE 2nd International Conference on Image and Graphics 2002[C], Hefei, China, 2002, **4675**: 1017 ~ 1021.

5 Kaufman A. An algorithm for 3D scan-conversion of polygons[A]. In: Proceedings of Eurographics Conference [C], NorthHolland, Amsterdam; Elsevier Science Publishers, 1987: 197 ~ 208.

6 Levoy M. A hybrid ray tracer for rendering polygon and volume data [J]. IEEE Transactions on Computer Graphics and Applications, 1990, **10**(2): 33 ~ 40.

7 Kreeger K A, Kaufman A E. Mixing translucent polygons with volumes[A]. In: Proceedings of Conference on Visualization '99 [C], San Francisco, CA, USA, 1999: 191 ~ 525.

8 Schmidt Ana Elisa F, Gattass Marcelo, Carvalho Paulo Cezar P. Combined 3D visualization of volume data and polygonal models using a Shear-Warp algorithm[J]. Computers & Graphics, 2000, **24**(4): 583 ~ 601.

9 YUAN Fei-niu, ZHU-GE Bin, ZHOU He-qin, *et al.* A fast volume rendering algorithm based on Intel SIMD and segmentation technologies. Journal of Image and Graphics, 2003, **8** (12): 1438 ~ 1443.[袁非牛, 诸葛斌, 周荷琴等. 基于奔腾 SIMD 和分割技术的快速体绘制[J]. 中国图象图形学报, 2003, **8**(12): 1438 ~ 1443.]

10 Wan Ming, Bryson Steve, Kaufman Arie. Boundary cell-based acceleration for volume ray casting[J]. Computer & Graphics, 1998, **22**(6): 715 ~ 721.

11 Kruger J, Westermann R. Acceleration techniques for GPU-based volume rendering [A]. In: Proceedings of IEEE Conference on Visualization[C], Seattle, Washington, USA, 2003: 287 ~ 292.