中图法分类号: TP391.41 **文献标志码**: A **文章编号**: 1006-8961(2010)12-1826-07
论文索引信息: 邹永宁, 王珏, 黄霞. 基于 GPU 的工业 CT 体数据剖切显示[J]. 中国图象图形学报,2010,15(12):1826-1832

基于 GPU 的工业 CT 体数据剖切显示

邹永宁, 王 珏, 黄 霞
(重庆大学光电技术及系统教育部重点实验室 ICT 研究中心,重庆 400030)

摘 要: 为了优化工业 CT 体数据的 3 维显示性能,必须提高剖切显示的速度,为此可将基于图形处理器(GPU)的编程方法应用到体数据显示算法中来,由于 GPU 可利用可编程流水线来加速渲染复杂场景,而且图形应用程序可调用片段着色程序来实现剖切显示算法,这样几十个可编程片段处理器可同时对多个像素点进行采样。为了提高剖切显示速度,提出了一种基于 GPU 的 CT 体数据剖切显示算法,并首先阐述了如何加载和预处理工业 CT 体数据,然后给出了 Cg 应用程序和片段着色程序的设计步骤;最后对工业 CT 扫描体数据进行了剖切实验。实验结果表明,基于 GPU 的剖切显示算法不仅生成的图像清晰,具有稳定和较高的帧率,而且计算速度是 CPU 算法的 2 ~ 9 倍,可满足了工业 CT 图像系统快速 3 维显示的要求。

关键词: 工业 CT; 图形处理器; 剖切显示; 硬件加速

Slice display of industrial CT volume data based on GPU

ZOU Yongning, WANG Jue, HUANG Xia
(*ICT Research Center, Key Laboratory of Optoelectronic Technology and System of the Education Ministry of China, Chongqing University, Chongqing* 400030)

Abstract: In order to optimize 3D display performance for volume data of industrial CT, it is necessary to accelerate slice display. Programming method based on graphics processing unit (GPU) is applied to volume display algorithm. Because GPU render complex scenes using a programmable parallel pipeline and slice display algorithm is implemented by graphics application which call fragment shading program, many programmable fragment processors can contemporaneously sample for lots of pixels. A slice display algorithm of CT volume based on GPU is presented. Firstly loading and preprocessing industrial CT volume data are interpreted. Then design steps of Cg application and fragment shading program are presented. Experiments of cutting muli-slice industrial CT images is carried out, the experimental results show that slice image generated by GPU algorithm is clear, the frame rate is steady and high, speed of computing is 2 ~ 9 times than CPU algorithm's. It meets the needs of industrial CT image system in terms of rapid 3D display.

Keywords: industrial CT; graphics processing unit(GPU); slice display; hardware acceleration

0 引 言

工业计算机层析成像(CT)在对被检测的物体进行锥束扫描或连续断层扫描之后,即可以获得 CT 体数据,然后通过体可视化技术就可以展现物体虚拟的 3 维结构与内部缺陷,使工程技术人员能够从不同的视角和不同的视觉效果观察物体每个部位的状态,并作出准确的测量和评判。工业 CT 图像的体可视化技术主要包括:面绘制、直接体绘制和剖切显示[1]。本文讨论的是剖切显示,已经有许多学者在这方面做了研究,传统的方法是利用中央处理器

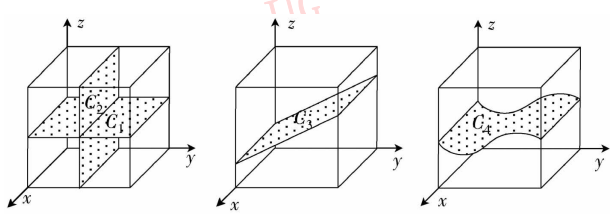
基金项目:国家高技术研究发展计划(863)基金项目(2006AA04Z104)。
收稿日期:2009-03-25;**改回日期**:2009-09-09

第一作者简介:邹永宁(1974—),男。讲师。重庆大学光电工程学院博士研究生。主要研究方向信息获取与处理、工业 CT 图像重建与显示、高性能计算机的并行计算等方面的研究。E-mail:zynlxu@sina.com。

(CPU)进行计算,卢艳平等人提出的一种切片重组方法,可连续自动地获取任意方向的序列切片图像[2]。张爱东等人用 Matlab 实现了对工业 CT 连续图像的剖切和透明显示[3]。王卫红等人提出了一种平面剖切、立体开窗及任意交互切割的算法[4]。王立功等人实现了对医学图像体数据场的 2 维和 3 维剖切显示[5]。近年来,图形处理器(GPU)发展迅速,许多学者应用 GPU 来解决世界性的高强度计算问题。Peter Kehoe 等人测试了 GPU 在自动构建视频,尤其在镜头边界检测和关键帧选择方面的潜力[6]。Hiroyuki 等人在 GPU 上通过运行,改进了计算精度的辐射热传递仿真程序,使 GPU 极大地加速了外形因子的计算[7]。Ondrej Fialka 等人研究了图像滤波算法的加速技术,在 GPU 上实现了快速傅里叶变换和卷积算法,并发现它们的执行速度远快于 CPU 的速度[8]。根据上述研究成果推断,如果直接利用图形硬件来加速剖切显示算法,也会得到意想不到的效果,基于此设想,本文研究并实现了直接应用 GPU 的加速剖切显示算法,最后与 CPU 算法进行了比较分析。

1　剖切显示算法研究

体数据场的 2 维剖切显示可分为正交剖切显示、任意斜面剖切显示,以及任意曲面剖切显示 3 种方式,图 1 显示了数据场中的 3 种剖切方式和虚拟剖切平面,图 1 (a)(b)中的 C_1、C_2、C_3 是剖切平面,图 1 (c)中的 C_4是剖切曲面[6]。正交剖切方式的算法较容易,实际应用较多,由于任意斜面的剖切算法较难,而任意曲面的剖切算法则更难,因此应用不多。



(a) 正交剖切　　(b) 任意斜面剖切　　(b) 任意曲面剖切

图 1　体数据场的 3 种剖切显示方式

Fig. 1　Three modes of slice display in volume data

工业 CT 断层序列图片是 3 维坐标系中的规则数据场,在剖切显示前先要定义 3 维空间坐标,通常可把重建的坐标系作为可视化的坐标系,即把扫描转台的转轴线作为 Z 轴,在断层图像平面定义 X 和 Y 轴。剖切显示就是先在体数据中定义一个虚拟的平面,然后再通过光栅化虚拟平面来得到 N^2 个坐标(N 为图像的宽度),同时重新采样每个坐标上的灰度值。

重采样过程需要两个步骤,即首先求出剖面上各像素在原始数据中的插值坐标,然后根据插值坐标进行插值计算得到该点的灰度值,一般采用三线性插值的效果较好。当算出剖面上所有像素点的灰度值后,即可得到剖面图像。3 维图像的任意位置的剖切显示流程图 2 所示。

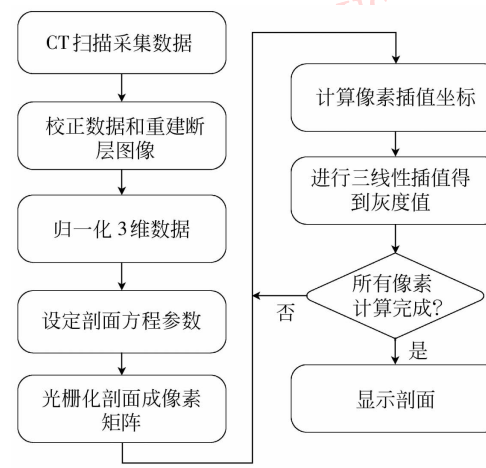


图 2　剖切显示流程图

Fig. 2　Flow chart of slice display

2　GPU 图形硬件编程

随着图形处理器最近几年的迅速发展,国内外学者已经将基于 GPU 的通用计算作为一个新的研究领域。而随着 GPU 的发展,其编程语言也在发展着,从最初的汇编语言发展到了现在的 Cg(C for graphics)语言、HLSL 语言和 GLSL 语言。如今,GPU 仍在高速发展,它的编程语言也在不断地完善,最近 nVidia 公司推出了 CUDA 编程环境,使基于 GPU 的通用计算编程变得越来越简单。

GPU 是按照图形硬件流水线的形式进行设计的,而且 GPU 内提供了可编程部件,例如图 3 是一个可编程图形处理器流水线中的顶点处理和片段处理阶段的示意图[9]。这种顶点和片段处理是可编程单元。可编程顶点处理器是一个硬件单元,可以运行 Cg 顶点程序,而可编程片段处理器则是一个

可以运行 Cg 片段程序的单元。由于 GPU 内部集成了许多流处理器,如 G80 系列 GPU 内具有 128 个流处理器,而这些处理器可以并行执行渲染程序,因此会带来极高的加速比。

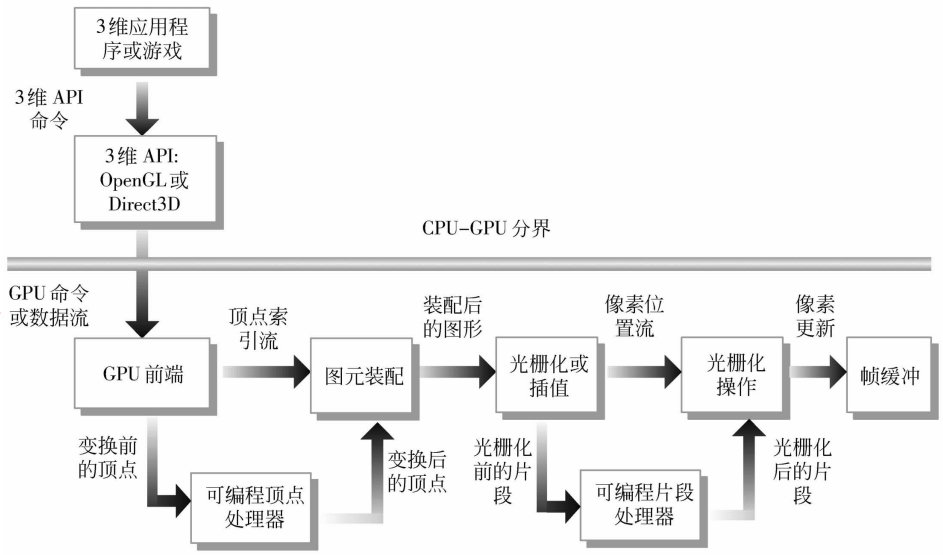


图 3 可编程图形流水线

Fig. 3 The programmable graphics pipeline

顶点着色程序主要完成图形顶点从物体空间到裁剪空间的变换,并将光线位置向量和视点位置向量变换到物体局部空间,在完成上述操作之后,再将纹理和光照计算所需的单位向量送入可编程片段着色器;片段处理器的只读寄存器包含了片段图元的顶点参数经过插值获得的片段参数,但不是顶点属性,片段着色程序先通过这些插值参数来计算纹理的坐标,再调用纹理采样函数获取颜色值。

Cg 只是利用可编程图形处理器来实时渲染复杂的 3 维场景的全部软件和硬件基础结构的一个组成部分,Cg 程序还需要依赖于 OpenGL 或者 Direct3D 这两种标准的 3 维编程接口来对图形处理器传达渲染命令。本文采用 OpenGL 环境编程接口[10]来传达指令。通常 Cg 源程序不能直接在 GPU 上执行,需要编译成可执行的代码,但 Cg 和一般高级语言不同,因为它支持动态编译,Cg 的应用程序可通过调用 Cg 运行库程序(通常以字母"cg"作为前缀)来编译和操作 Cg 程序,所以应用程序不需要重编译就可以直接执行修改后的 Cg 源程序(保证接口不变)。

3 体数据场剖切显示的具体实现

3.1 数据加载和预处理

加载的数据可以是断层序列图像,也可以是一个完整的 3 维体数据文件。本文试验用的数据为工业 CT 连续扫描的断层序列图像。加载时,首先按顺序依次将所有图像加载到 1 维数组中。存储的坐标顺序为 X、Y、Z,即 X 表示图像的列号,Y 表示图像的行号,Z 表示图像的层号。数组每个元素占 1Byte,数组的长度为 $M \times M \times N$(其中 M、N 分别为单层图像的宽度和所有图像的层数),由于重建的断层图像是正方形,因此单层图像的宽度和高度相等,只需要用一个参数表示即可。假设任意体素的空间位置为(X,Y,Z),则其在数组中的位置为

$$L = Z \times (M \times M) + Y \times M + X$$

相反,由体素在数组中的位置 L,亦可求出其空间坐标(X,Y,Z)。

$$X = [L \bmod (M \times M)] \bmod (M)$$

$$Y = [L \bmod (M \times M)]/M$$

$$Z = L/(M \times M)$$

式中,mod 为取余数操作符。

这样 3 维空间坐标和 1 维存储数组就建立了对应关系,即只要知道体素的 3 维空间坐标就可以计算出体素在内存中的位置,也就知道体素的灰度值。

为了使图像的质量更好,就要进行数据预处理,主要包括图像增强、去噪和 3 维数据层间归一化,一般使用灰度拉伸、直方图均衡化即可以增强图像,可用中值滤波和高斯滤波去噪。由于不同层图像的扫

描时段不同,致使产生层间不一致性,因此需要进行层间归一化。

3.2　任意平面剖切显示方法

正交剖切是任意平面剖切显示的一种特例,因此弄清楚任意平面剖切显示方法,也就容易导出正交剖切显示方法了。如图 4 建立的数据场空间 3 维直角坐标系,坐标原点为 $\boldsymbol{O}$,$\overrightarrow{OP}$垂直于剖切平面 π,且点 $\boldsymbol{P}$ 在剖切平面内,故$\overrightarrow{OP}$是剖切平面的法向量。这样只要给定剖切平面中心点 $\boldsymbol{P}$ 的坐标和原点到平面的距离 $|\overrightarrow{OP}|$ 就可以唯一确定剖切平面方程。由于剖切平面需要光栅化为有限区域像素矩阵才能显示,故假设 $ABCD$ 是剖切平面上将要光栅化的矩形区域,为了简化算法,可将矩形 $ABCD$ 的中心设在点 $\boldsymbol{P}$,视点(眼睛)位于$\overrightarrow{OP}$的延长线上的无穷远处,那么线段 AB 的长度等于剖切图像的宽,AD 等于剖切图像的高。光栅化区域中每个像素点的空间坐标可以通过剖切平面方程和 $ABCD$ 的位置经过计算得到。

3 维空间中任意平面由法向量和原点到平面的距离唯一确定,设剖切平面 π 的单位法向量为 $\boldsymbol{n}=(n_x,n_y,n_z)$,则 $\boldsymbol{n}=\dfrac{\overrightarrow{OP}}{|\overrightarrow{OP}|}$,设 $|\overrightarrow{OP}|=p$,剖切平面 π 的方程可写成

$$n_x x + n_y y + n_z z - p = 0 \tag{1}$$

设 $ABCD$ 矩形区域中任一光栅化像素点为 $\boldsymbol{Q}$,则

$$\overrightarrow{OQ} = \overrightarrow{OP} + \overrightarrow{PQ} \tag{2}$$

设 uv 坐标轴上的单位向量分别为 $\boldsymbol{u}$ 和 $\boldsymbol{v}$,$\boldsymbol{Q}$ 在 uv 平面上的投影为点 $\boldsymbol{I}$ 和点 $\boldsymbol{J}$,$|\overrightarrow{PI}|=i$,$|\overrightarrow{PJ}|=j$,即像素点在 uv 坐标系中的位置为(i,j),其中 $-M/2<i<M/2$,$-M/2<j<M/2$,则有

$$\overrightarrow{PI} = i\boldsymbol{v},\ \overrightarrow{PJ} = j\boldsymbol{u}$$

又$\overrightarrow{PQ}=\overrightarrow{PI}+\overrightarrow{PJ}$,代入式(2),有

$$\overrightarrow{OQ} = \overrightarrow{OP} + \overrightarrow{PI} + \overrightarrow{PJ} = \overrightarrow{OP} + i\boldsymbol{u} + j\boldsymbol{v} \tag{3}$$

由此可见,只要给出点 $\boldsymbol{P}$ 的坐标和向量 $\boldsymbol{u}$,$\boldsymbol{v}$ 就可以求出矩阵内所有像素点的空间坐标。前面讲过,正交剖切是任意剖切的特例,从图 4 来看,就是点 $\boldsymbol{P}$ 在 x、y 或 z 轴上,且向量 $\boldsymbol{u}$,$\boldsymbol{v}$ 与坐标轴平行,式(3)相应变成

$$\overrightarrow{OQ} = i\boldsymbol{x} + j\boldsymbol{y} + k\boldsymbol{z}$$

其中 $\boldsymbol{xyz}$ 分别是 xyz 轴的单位向量。

假设求得数据场中采样点坐标为(x,y,z),则取采样点周围 8 个体素的灰度值进行三线性插值即得到采样点的灰度值。

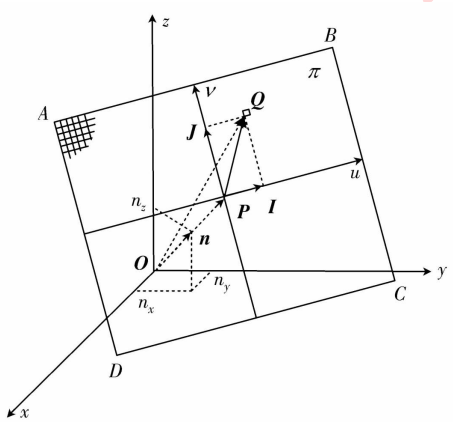


图 4　剖切几何结构图

Fig. 4　Geometry structure of slice

3.3　应用程序的设计

前面讲过,由于 Cg 的图形功能要依赖于实时 3 维的 OpenGL 和 Direct3D 编程接口,因此基于 GPU 的剖切显示算法要按照图形算法流程来进行编写,即首先需要编写实时 3 维图形应用程序,本文采用 OpenGL 编程接口编写应用程序;其次需要编写 Cg 着色程序。OpenGL 应用程序的步骤如下:

1) 初始化 OpenGL 环境,设置窗口参数。初始化 Cg 扩展环境,并指定入口函数的名字和编译入口函数所需要的 profile 的名字,由于本文算法只是生成一个简单的四边形,所以只需指定片段着色程序即可,而不需要顶点着色程序;

2) 加载 3 维体数据文件,生成纹理名称和绑定纹理对象,并定义纹理属性、图像属性和纹理数据;

3) 获取 Cg 程序接口中的 uniform 参数(如采样对象、视角等),为采样对象设置已创建的纹理对象;

4) 设置显示回调函数,启动运行程序显示回调函数包括绑定片段着色器,设置 uniform 参数,绘制一个带贴图的正四边形。

3.4　片段着色程序设计

根据第 1 节已介绍的 GPU 流水线处理过程,本文应用程序中所画的四边形,经过顶点处理、图元装配和光栅化,即生成了 $M\times M$ 个片段,每个片段有各自不同的属性,本文算法主要是先使用片段的纹理坐标属性,所谓纹理坐标实质就是四边形中每个

像素点的坐标;然后可编程片段处理器载入片段着色程序指令,逐个处理片段;最后生成像素颜色,并输出显示。由于 GPU 内有很多个片段处理器可以并行计算,因此 GPU 的整体运行速度远比 CPU 高。片段着色程序入口函数名称在应用程序中已经指定,本文中把 main 作为入口函数,输入的片段属性参数为纹理坐标(texCoord),输出 color,uniform 参数有 3 维纹理 decal 和倾斜角度 θ,以及剖切平面的深度 $d(0\leqslant d\leqslant 1.0)$。片段着色程序步骤如下:

1)若把 3 维坐标原点平移到纹理的正中心,则 $d-0.5$ 是采样点的 y 轴坐标,2 维纹理坐标向量 $(x_{\text{texture}},y_{\text{texture}})$减去(0.5, 0.5)之后分别就是采样点在 x 轴和 z 轴的坐标,这样即可求得初始采样点在新坐标系中的值;

2)将采样点坐标旋转角度 θ,即得到实际剖切采样点的坐标;

3)将实际采样点的坐标平移到纹理坐标系,即可得到 3 维纹理坐标,再调用采样函数 tex3D 进行采样,即得到采样点的 RGB 颜色值,然后将采样点颜色赋值给片段颜色值,返回。

以上步骤实现了一个平行 x 轴的剖切平面在 3 维数据场的采样,纹理采样函数 tex3D 属于 Cg 标准库函数,它可根据纹理坐标指定的位置采样数据。如果对 3 维纹理属性 GL_TEXTURE_MIN_FILTER 和 GL_TEXTURE_MAG_FILTER 指定参数 GL_LINEAR,那么 GPU 就会自动执行对一块 2×2×2 纹理单元进行加权平均运算。这样既省去了三线性插值编程,又提高了采样速度。

应用程序通过传递不同的深度或角度参数就可以实现剖切平面的平移或旋转。如果 Cg 程序中增加一个角度参数,则可以使剖面在 3 维空间任意旋转。如果想输出伪彩色,就可以在入口参数中添加一维采样对象作为颜色传递函数,同时应用程序中要创建一个 1 维纹理与之对应,最后片段程序再添加一行 tex1D 函数调用即可。

4 实验与结果

本文所采用的实验平台是 2.93GHz 的 PC 机,内存为 1GB,Geforce790GS 显卡,显存为 512M。图 5 是通过 CD-650BX 型 CT 机扫描重建的化油器断层序列图像,一共有 167 张 512×512 pixel 大小的位图(只显示了其中 8 张)。

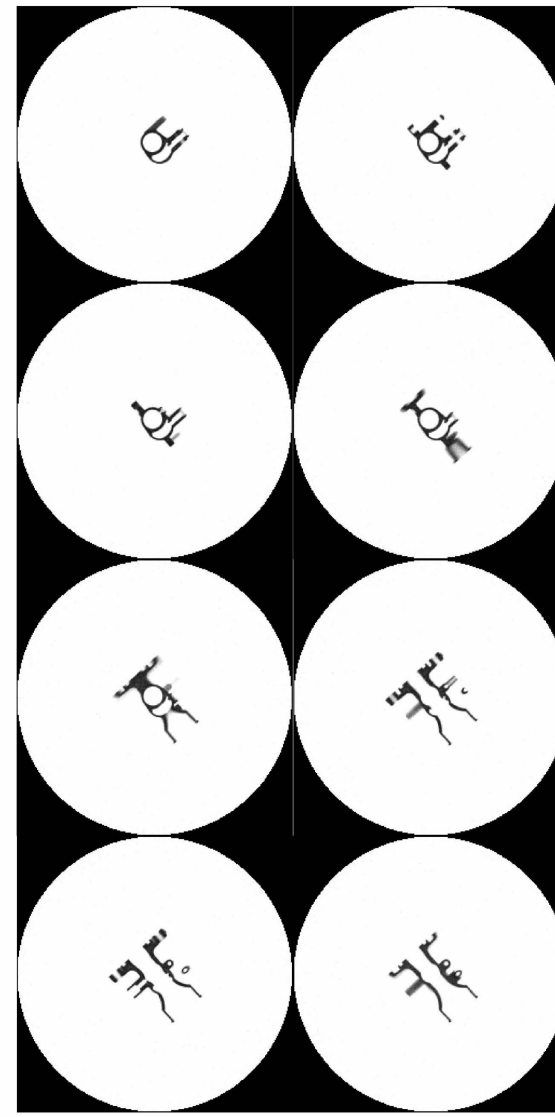


图 5 连续断层扫描序列图片

Fig. 5 Serial images of continuous scan of tomography

运行应用程序,当给角度参数 θ 输入 0°,则等价于正交剖切,图 6 显示了正交剖切的 6 幅图像,当深度 d 取不同值,则可以获得同一法向不同位置的剖面图。当角度 θ 不等于 0°时,就是斜面剖切,图 7 显示斜面剖切的 4 幅图像,θ 取 45°,深度 d 分别取不同的值,此时片段程序接收到的纹理坐标是垂直于坐标轴剖面的坐标,需经过旋转变换之后才是实际的采样坐标。实验结果表明,利用 GPU 实现的剖切显示可以获得清晰的剖面图像。

为了比较 GPU 和 CPU 的运行速度,笔者在相同配置的机器上运行了基于 VTK 编程的剖切程序。VTK 程序是通过 CPU 进行采样计算的,无图形硬件

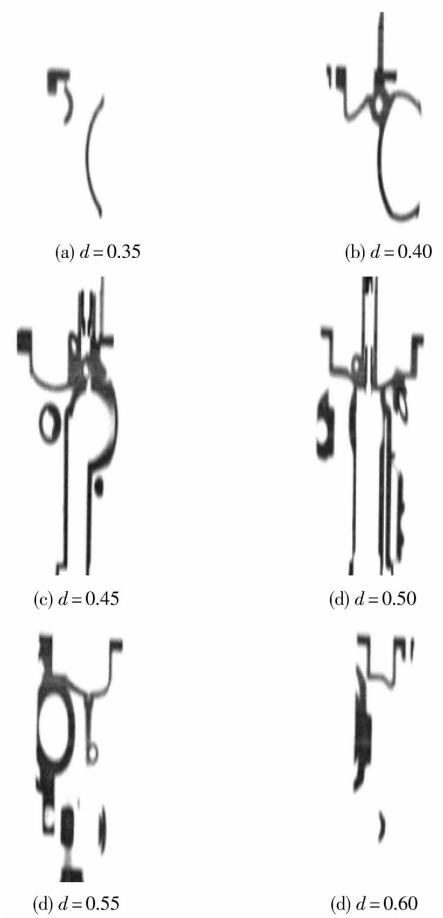


图 6　正交剖切显示图(θ 等于 0°)

Fig. 6　Orthogonal slice images, θ equal to zero

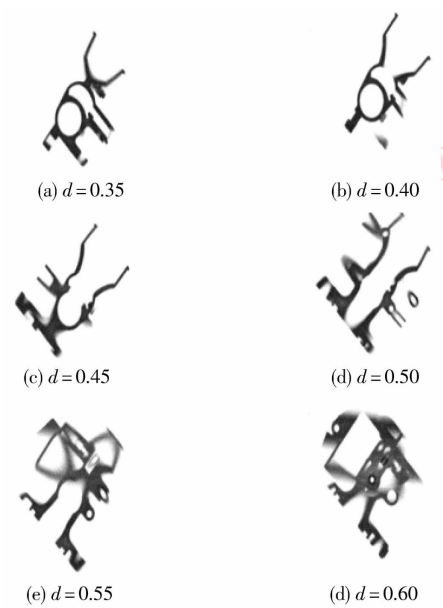


图 7　斜面剖切显示图(θ 为 45°)

Fig. 7　Inclined slice images, θ equal to 45°

加速。两种程序的运算速度比较见表 1,分析表中数据可知,无论是小数据场还是大数据场,GPU 都表现出了稳定和较高的帧率,因此可推断出数据量不是影响 GPU 运行速度的瓶颈,也就是说,GPU 非常适宜用于处理超大规模数据,而 CPU 的帧率则随着数据场的增大而减小。显然 GPU 加速的算法在绘制速度上具有很大的优势。

表 1　GPU 和 CPU 剖切显示算法实验结果

Tab. 1　Results of slice display algorithm on GPU and CPU

体数据场大小	GPU 算法帧率/(帧/s)		帧率比
	(1)	(2)	(1)/(2)
$256^2\times100$	72	35	2.0
$512^2\times100$	72	26	2.7
$1\,024^2\times40$	72	8	9.0

5　结　论

本文讨论了一种工业 CT 序列图像的剖切显示方法,这种方法区别于传统的使用 CPU 来进行采样,而是利用 GPU 的高密度并行计算和运用 Cg 图形编程方法来实现快速准确的采样,并可任意调整剖切面的方向和位置。实验结果显示,使用 GPU 算法剖切体数据的帧率是 CPU 算法的 2 ~ 9 倍,这对于增强工业 CT 图像软件性能具有重要意义。由于 GPU 显卡的应用逐渐普及,所以这种方法可以推广到普通的 PC 计算机上实现。工业 CT 图像可视化方法还包括面绘制和体绘制,因此如何应用 GPU 对其他显示方法进行加速即成为需要进一步研究的问题。

参考文献(References)

[1] Tang Zesheng. Visualization of 3D Data Field [M]. Beijing: Tsinghua University Press, 1999. [唐泽圣. 三维数据场可视化[M]. 北京:清华大学出版社,1999.]

[2] Lu Yangping, Wang Jue, Liu Rong. Study on reslicing method for volume data from industrial CT [J]. Computer Engineering and Applications, 2007,43 (22):201- 203. [卢艳平,王珏,刘荣. 工业 CT 体数据切片重组方法研究[J]. 计算机工程与应用,2007,43(22):201-203.]

[3] Zhang Aidong, Li Ju, Sun Lingxia. Three dimensional reconstruction of continuous ICT images by MATLAB [J]. Nuclear Electronics & Detection Technology, 2006,26(4):489-491. [张爱东,李炬,孙灵霞. MATLAB 编程实现连续断层工业 CT 图像的三维重建[J]. 核电子学与探测技,2006,26(4):

489-491.]

[4] Wang Weihong, Qin Xujia, Zheng Hongbo. Virtual cutting algorithms for 3D surface model reconstructed from medical images [J]. Journal of Image and Graphics, 2006,11(2):217-223.[王卫红,秦绪佳,郑红波. 医学图像 3 维重建模型的虚拟剖切算法[J]. 中国图象图形学报, 2006,11(2):217-223.]

[5] Wang Ligong, Yu Yonghua, Luo Limin. The section display method for medical image volume Data set [J]. Journal of Image and Graphics, 2003,8(A):769-773.[王立功,于甬华,罗立民. 医学图象体数据场的剖切显示方法[J]. 中国图象图形学报, 2003,8(A):769-773.]

[6] Kehoe Peter, Smeaton Alan F. Using graphics processor units (GPUs) for automatic video structuring [C]//Proceedings of Eight International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07). Los Alamitos, CA, USA: IEEE Computer Society Press, 2007:18-22.

[7] Hiroyuki Takizawa. Radiative heat transfer simulation using programmable graphics hardware [C]//Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering. Los Alamitos, CA, USA:IEEE Computer Society Press, 2006:29-37.

[8] Ondrej Fialka, Martin Cadik. FFT and Convolution Performance in Image Filtering on GPU[C]//Proceedings of the Information Visualization (IV'06). Los Alamitos,CA,USA:IEEE Computer Society, 2006:609-614.

[9] Randima Fernando, Kilgard Mark J. The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics [M]. Boston, Massachusetts, USA: Addison-Wesley, 2003:32-45.

[10] Dave Shreiner, Mason Woo, Jakie Neider, et al. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2[M]. 5th ed. Boston, Massachusetts, USA: Addison-Wesley Press, 2005:63-96.