徐赛花, 张二华. 基于 CUDA 的三维数据并行可视化[J]. CT 理论与应用研究, 2011, 20(1): 47-54.
Xu SH, Zhang EH. CUDA-based parallel visualization of 3D data[J]. CT Theory and Applications, 2011, 20(1): 47-54.

基于 CUDA 的三维数据并行可视化

徐赛花，张二华✉

（南京理工大学计算机科学与技术学院，南京 210094）

摘要：科学计算可视化是发达国家 20 世纪 80 年代后期提出并发展起来的一个新的研究领域，它运用计算机图形学和图像处理技术，将三维数据转换为图形及图像在屏幕上显示出来并进行交互处理，主要包括面绘制和体绘制两种方法。光线投射算法是最经典的三维数据体绘制方法，图像质量较高，但计算时间较长，基于 CUDA 的编程技术利用 GPU 的多核并行运算功能可显著提高计算速度，实现大规模三维数据的实时可视化。

关键词：可视化；光线投射算法；GPU；CUDA

文章编号：1004-4140（2011）01-0047-08 **中图分类号**：P 631；TP 301.6 **文献标识码**：A

科学计算可视化自提出以来已经历了 20 多年的发展，在科学研究、医学、石油勘探、工业无损探测等领域得到了广泛应用。可视化可分为面绘制和体绘制两种方法[1-2]，面绘制首先由三维数据构造出中间几何面（如物体外表面或等值面），再由传统的计算机图形学技术绘制这些曲面，这种方法得到的可视化图像不能反映整个数据的全貌及细节。体绘制方法不需要构造中间几何面，直接由三维数据生成屏幕上的二维图像，能反映三维数据的全貌和细节，并具有图像质量高、便于并行处理的优点，其缺点是计算量很大。

基于软件的体绘制方法由于计算量大，对大规模三维数据难以达到实时绘制的要求，随着图形硬件技术的迅速发展，体绘制也开始寻求基于图形硬件的加速方法。图形处理器（Graphics Processing Unit，GPU）的出现为实时体绘制提供了硬件支持，把原本由 CPU 执行的计算和绘制工作转移到 GPU 上，大大减轻了 CPU 的负担，GPU 独有的可编程管线技术极大地提高了计算速度，同时保持了原有的图像质量。随着 GPU 的不断发展，相继提出了 GLSL、HLSL、CG 等高级着色语言，进一步提高了 GPU 的可编程性，基于 GPU 的体绘制方法也得到了进一步发展。随着 GPU 可编程性的不断提高，利用 GPU 完成通用计算的研究渐渐活跃起来，被称为 GPU 的通用计算（General Purpose of GPU，GPGPU）[3]，并逐渐发展为一个新的研究方向。

2007 年，nVidia 公司提出了计算统一设备架构（Compute Unified Device Architecture，CUDA）编程模型的概念[4]，其特点是不需要借助图形学应用程序编程接口（Application Programming Interface，API）就可以通过类 C 的高级语言直接操作 GPU 的硬件资源，相比其他高级着色语言更易于理解和掌握，其特有的 Grid、Block、Thread

收稿日期：2010-12-03。

三层结构更利于并行处理。由于在性能、成本和开发时间上都有显著优势，CUDA 概念一经推出便在学术界和产业界引起强烈反响。2008 年各种支持 CUDA 编程模型的显卡相继面世，目前 CUDA 已在众多领域获得应用，并取得了丰硕的成果。在三维数据可视化方面，目前国内外对 CUDA 技术的研究还处于起步阶段，但发展势头非常强劲，相信不久的将来 CUDA 技术将完全融入到可视化行业。

本文的主要目的是对传统的光线投射算法进行加速，采用的是 GPU 中的 CUDA 编程模型。使用 CUDA 编程技术利用 GPU 的多核并行运算功能可以很大程度上提高计算速度，同时又避免了传统 GPU 算法中繁琐的图形 API。本文利用 CUDA 实现了基于光线投射算法的三维数据可视化，并与传统的软件方法进行了对比分析，得到了理想的结果。

1 光线投射体绘制算法

20 多年来人们先后提出了多种体绘制方法，其中光线投射（ray-casting）算法计算过程最明确，可视化图像的精度较高，得到了广泛的应用。

光线投射算法的基本原理[1, 5-7]是，先根据转换函数将数据赋予相应的颜色和不透明度值，从屏幕上的每一个像素点出发沿视线的方向发出一条射线，这条射线穿过三维数据体，再在这条射线上等间距设置一系列重采样点，由距离重采样点最近的 8 个原始数据点通过三线性插值，求出重采样点的颜色和不透明度。然后，按从前至后或从后至前的顺序对重采样点的颜色和不透明度进行合成，得到该像素点的颜色，将所有像素点的颜色拼接起来就得到一幅完整的可视化图像。具体步骤可描述如下：

```
for y=1 to W        /*W 为屏幕宽度*/
   for x=1 to H     /*H 为屏幕高度*/
      Opacity=0.0;        /*不透明度置初值 0*/
      Image[y][x]=BackGroundColor（背景颜色）; /*像素颜色初值为背景色*/
      计算像素点(x,y)对应的物体空间坐标 P(x,y,z);
      从像素点发射一条射线 Q=P+v⃗t，其中 v⃗ 表示视线方向，用数据体包围盒
      对射线进行剪裁，求射线射入数据场和离开数据场的参数 t1，t2;
      While t1≤t≤t2 && Opacity<1.0
          当前射线上采样点坐标为 SamP(x,y,z)=P+v⃗t，用三线性插值求当前
          采样点的颜色值 Cnow;
          累积不透明度值，更新 Opacity;
          进行图像合成，更新 Image[y][x];
      End
   End
End
```

2 GPU 通用计算

GPU 的发展经历了固定管线、可编程 GPU、GPU 通用计算、CUDA 等多个阶段。

早期的 GPU 通过流水线来完成 3D 模型到图像的渲染工作。流水线输入的是三维物体的描述，包括顶点坐标、法线方向、物体表面材质、光源所在位置等。渲染过程被分为几

个可以并行处理的阶段，分别由流水线的不同单元进行处理。在 GPU 渲染流水线的不同阶段，需要处理的对象包括顶点、几何图元、片元和像素。典型的渲染过程可以分为顶点生成、顶点处理、图元生成、图元处理、片元生成、片元处理、像素操作等几个阶段。最初的 GPU 不具备可编程性，只能通过固定管线完成渲染工作，功能单一。可编程 GPU 用可编程的顶点处理器（vertex processor）和片元处理器（fragment processor）取代了固定管线的顶点处理部分和像素处理部分。可编程 GPU 上的编程语言主要有 Cg、高级着色器语言（High Level Shader Language，HLSL）、OpenGL 着色语言（OpenGL Shading Language，GLSL）3 种，它们的特点是需要借助图形 API，而图形 API 复杂繁琐，对非专业人员往往很难熟练掌握。

GPU 最大的优点在于超强的计算能力，很多研究者试图将其应用于非图形计算，2001 年 Larsen 在 GPU 上实现了矩阵运算。将 GPU 用于非图形的通用并行计算称为 GPGPU。2007 年以前，GPGPU 程序设计仍然难以摆脱图形 API 的繁杂接口，影响了该技术的进一步应用，并且图形 API 的 SIMD（单指令多数据）模式不利于存储器的灵活操作，也使 GPGPU 的应用范围受限。为解决这些问题，nVidia 公司推出了 CUDA 编程模型。CUDA 是一种将 GPU 作为数据并行计算设备的软硬件体系，CUDA 不需要借助图形学 API，而是采用比较容易掌握的类 C 语言进行开发，具有较灵活的存储器操作，易于使用。

3　CUDA 实现光线投射算法

3.1　可行性分析

CUDA 编程模型的核心是 Grid、Block、Thread 三层结构[4]，如图 1 所示，以线程网格（Grid）的形式组织，每个 Grid 由若干个线程块（Block）组成，而每个 Block 又由若干个线程（Thread）组成。程序以 Block 为执行单位，各 Block 并行执行，Block 之间无法通信，而同一个 Block 的 Thread 通过共享存储器来相互协作，通过合理安排对内存的访问来同步。Block 可以是一维、二维或者三维的，用内建变量 *threadIdx* 来标识各个线程。Grid 只能是一维或者二维的，用内建变量 *blockIdx* 来标识各线程块。每个 Block 中的 Thread 数目受到处理核心所拥有的存储器资源的限制，在 NVIDIA Tesla 架构中，一个 Block 最多可以包含 512 个 Thread[8−10]。

CUDA 编程模型将 CPU 作为主机（Host），GPU 作为设备（Device），CPU 与 GPU 协同工作，各司其职。CPU 负责进行逻辑性强的事务处理和串行计算，GPU 则专注于执行高度线程化的并行处理任务。CPU 和 GPU 各自拥有相互独立的存储空间。程序运行时，CPU 完成串行代码的执行后将并行计算的工作交给 GPU，运行在 GPU 上的 CUDA 并行计算函数称为内核函数（kernel），kernel 函数是整个 CUDA 程序中一个可以被并行执行的步骤。当调用 kernel 的时候，会有 N 个 CUDA 线程并行地执行 N 次，每个线程都有一个独特的线程 ID（*blockID* 和 *threadID*），用于与其他线程相区分，每个线程按照指令的顺序串行执行一次 kernel 函数，各线程之间是并行执行的。当程序运行到并行计算部分时，CPU 把数据传给 GPU，GPU 上所有线程同时执行 kernel 函数代码，等所有线程都计算完成后，GPU 再把计算结果传回 CPU。

光线投射算法中从各像素点发出的射线之间的计算过程是相互独立的，计算方式是相同的，具有高度的并行性[11]。把计算过程封装在 kernel 函数中，每条射线的计算任务由一

个 Thread 完成，把整个屏幕作为一个 Grid，屏幕上每个像素点作为一个 Thread，屏幕根据限制分为若干区域，每个区域作为一个 Block，充分利用了 CUDA 的三层结构，实现并行计算。

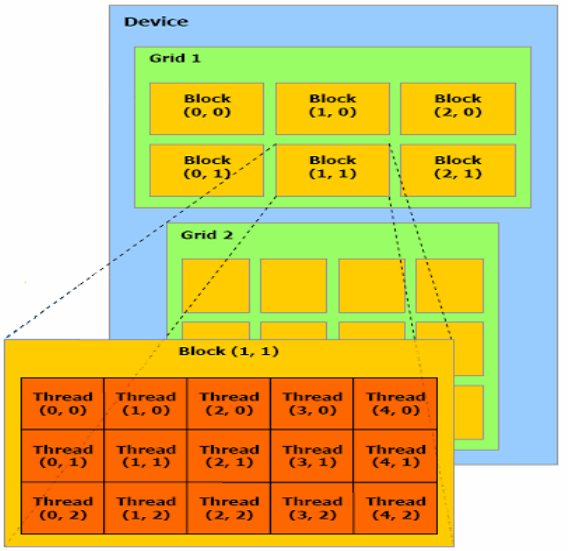

图 1 CUDA 的线程结构

Fig.1 The thread structure of CUDA

3.2 算法流程

CPU 上主要完成以下工作：在 CPU 上完成数据的文件读入，对数据进行分类，赋予颜色和不透明度，设置视线方向，初始化 OpenGL 环境。

GPU 上的算法流程描述如下：

第一步，把数据体、颜色转换函数和观察方向从 CPU 内存复制到 GPU 显存。

第二步，根据需要分配线程结构，调用 kernel 函数，计算屏幕像素颜色值。

第三步，把屏幕颜色值复制给 CPU，进行处理和绘制。

kernel 函数主要流程如下：

第一步，根据线程块和线程索引来计算对应的屏幕像素点的位置。

第二步，给屏幕像素的颜色和不透明度赋初值 0。

第三步，计算屏幕像素的物体空间坐标，并根据射线方向采用包围盒方法求出光线进入数据体和离开数据体的交点坐标。

第四步，从进入点开始，沿射线方向依次按预设的采样间距求重采样点坐标，并由该点周围的 8 个原始采样点用三次线性内插得到重采样点的颜色和不透明度值，再按从前向后方式进行累加合成，若屏幕像素点的不透明度值接近 1.0 则结束该光线的计算，否则继续取下一个重采样点，直至到达离开点为止。

第五步，把累积合成的屏幕像素的最终颜色值写入到显存空间，完成该像素的计算工作。

4　算法实验

4.1　实验环境

实验所用计算机 CPU 为 Intel（R） Pentium（R） Dual E2200，主频 2.20 GHz，1 GB 的内存，GPU 为 NVIDIA Quadro FX380，16 个 CUDA 并行处理器核心，256 MB GDDR3 显存，操作系统为 Windows XP Professional 2002。

实验所用数据包括 256×256×256 的医学脚掌三维 CT 数据、256×256×256 的新疆塔里木盆地三维地震数据和 251×387×67 的国外某地区三维地震数据。

4.2　实验结果

图 2 是三维脚掌 CT 数据分别，采用纯软件方法和 CUDA 方法得到的可视化图像。可以看出，用 CUDA 方法得到的可视化图像与用纯软件方法得到的图像基本相同，一些细微差别肉眼甚至看不出来，只有用图像软件进行分析才能找出差异之处。在计算时间方面，纯软件实现需要 2.313 s，而用 CUDA 实现只需 0.75 s，计算速度是原来的 3 倍。

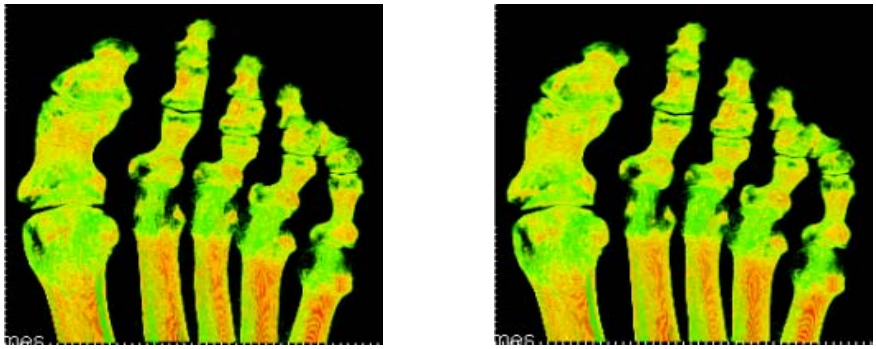

图 2　脚掌数据用纯软件方法实现（左）和用 CUDA 实现（右）

Fig.2　Foot data visualization by software (left) and CUDA (right)

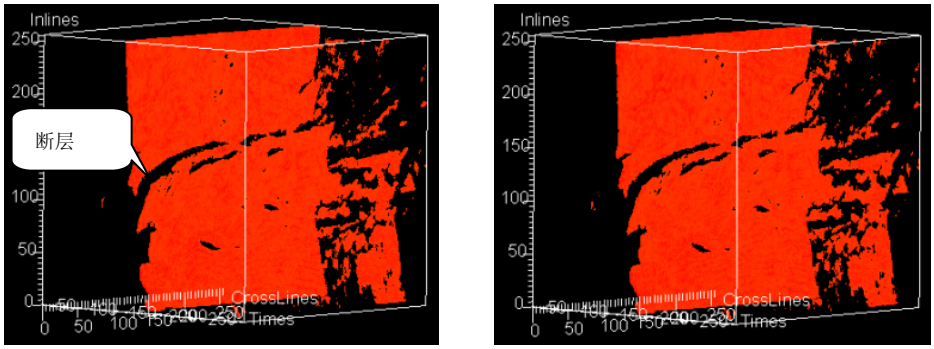

图 3　新疆塔里木盆地地震数据用纯软件实现（左）和用 CUDA 实现（右）

Fig.3　Seismic data visualization of Xinjiang's Tarim Basin by software (left) and CUDA (right)

图 3 是新疆塔里木盆地某地区 256×256×256 的实际三维地震数据可视化图像。两幅图都清晰地显示了断层的部位，纯软件实现需要 2.5 s，CUDA 实现只需 0.469 s，计算效率

是原来的 5.3 倍。

图 4 是国外某地区 251×387×67 的实际三维地震数据可视化图像。图中清晰地显示了断裂体系的分布，纯软件实现需 1.172 s，CUDA 实现只需 0.281 s，计算效率是原来的 4.2 倍。

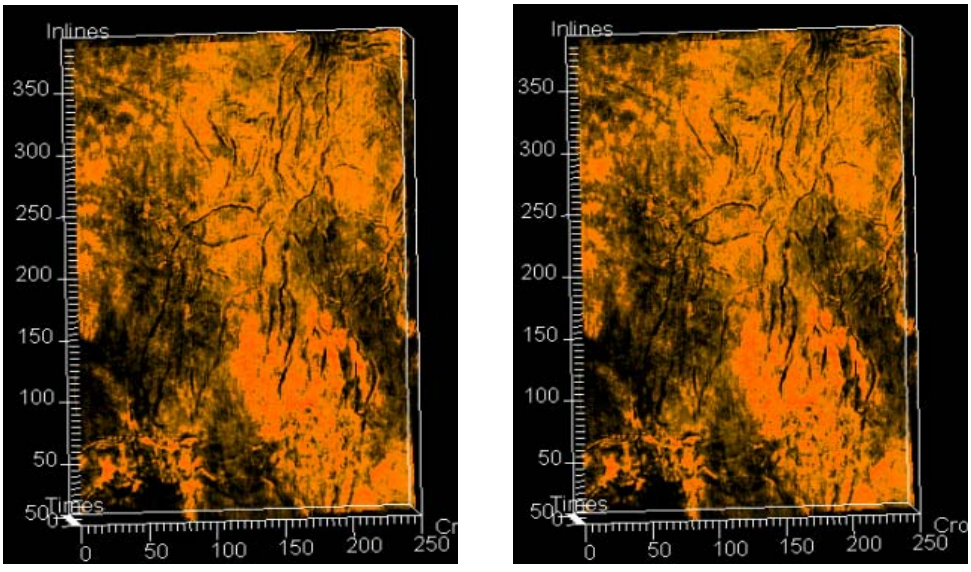

图 4 国外某地区地震数据用纯软件实现（左）和用 CUDA 实现（右）

Fig.4 Seismic data visualization of a foreign area by software(left) and CUDA(right)

表 1 列出了各组实验数据分别在两种方式下绘制所需时间，以及加速比。显然，用 CUDA 实现的光线投射算法在计算效率上有了显著提高，基本能满足实时交互显示的需要。从表 1 中可以看出不同数据体得到的加速比有一定差异，这主要是由数据体本身的结构、大小以及所选择的不透明度曲线决定。

表 1 绘制时间和加速比

Table 1 Rendering time and speedup ratio

数据体	实现时间/s		加速比
	CPU	CUDA	
256×256×256 的脚掌数据	2.313	0.750	3.08
256×256×256 的地震数据	2.500	0.469	5.33
251×387×67 的地震数据	1.172	0.281	4.17

5 结论

用 CUDA 实现的光线投射算法与纯软件实现相比，图像质量没有降低，而绘制速度有了明显提高。目前存在的问题是 GPU 显卡的显存有限，当数据过大时，不能全部加载进显存，若将数据体进行分块加载并分块进行计算，则各分块之间只能是串行的，势必会降低绘制

效率。另外，在实际工程计算中的程序远比本文的复杂，可能涉及多个并行过程，可以设计多个 kernel 函数，但若是一个 Thread 中存在并行过程，那就要涉及到 kernel 函数的嵌套问题了，目前的 CUDA 还不支持这一功能。但是 CUDA 还在发展中，相信不久的将来这些问题都会得到很好的解决。

参考文献

[1] 唐泽圣. 三维数据场可视化[M]. 北京：清华大学出版社, 1999.

[2] 张二华, 高林, 马仁安, 等. 三维地震数据可视化原理及方法[J]. CT 理论与应用研究, 2007, 16(3): 20-28.
Zhang EH, Gao L, Ma RA, et al. Principles and implementation of visualization on 3D seismic data volume[J]. CT Theory and Applications, 2007, 16(3): 20-28.

[3] 吴井胜, 鲍旭东. 基于 GPU 通用计算的图像体绘制[J]. 生物医学工程研究, 2008, 27(3): 175-178.
Wu JS, Bao XD. Image volume rendering based on GPU computing[J]. Journal of Biomedical Engineering Research, 2008, 27(3): 175-178.

[4] 张舒, 褚艳利. GPU 高性能运算之 CUDA[M]. 北京：中国水利水电出版社, 2009: 10.

[5] 孙薇薇. 光线投射体绘制算法关键技术研究[D]. 天津：天津理工大学, 2006.
Sun WW. Research on key technique of ray casting volume rendering algorithm[D]. Tianjin: Tianjin University of Technology, 2006.

[6] 李会欣. 体视化技术光线投射算法研究[D]. 北京：北京科技大学, 2006.
Li HX. The study of ray casting algorithm in volume visualization[D]. Beijing: University of Science and Technology Beijing, 2006.

[7] Danskin J, Hanrahan P. Fast algorithms for volume ray tracing[C]//Proceedings of Workshop on Volume Visualization, United States: Association for Computing Machinery, 1992: 91-98.

[8] 董现玲, 江贵平, 张煜. 基于 CUDA 的快速光线投射法[J]. 北京生物医学工程, 2010, 29(2): 125-129.
Dong XL, Jiang GP, Zhang Y. Accelerated ray casting algorithm based on cuda[J]. Beijing Biomedical Engineering, 2010, 29(2): 125-129.

[9] Marsalek L, Hauber A, Slusallek P. High-speed volume ray casting with CUDA[C]//IEEE Symposium on Interactive Ray Tracing, 2008: 185.

[10] 毕文元, 陈志强, 张丽, 等. 基于 CUDA 的三维重建过程实时可视化方法[J]. CT 理论与应用研究, 2010, 19(2): 1-8.
Bi WY, Chen ZQ, Zhang L, et al. Real-time visualize the 3D reconstruction procedure using CUDA[J]. CT Theory and Applications, 2010, 19(2): 1-8.

[11] 刘伟峰, 唐先明, 韩宝东, 等. 基于 GPU 计算的光线投射算法体绘制研究[C]//北京图像图形学学会. 图像图形技术研究与应用 2009：第四届图像图形技术与应用学术会议论文集, 北京：中国传媒大学出版社, 2009: 272-277.
Liu WF, Tang XM, Han BD, et al. Study on ray-casting-based volume rendering using GPU computing[C]//Beijing Institute of Image and Graphics. Study and Application of Image and Graphics Technology 2009: Proceedings of the 4th Image and Graphics Technology and Application Conference, Beijing: Communication University of China Press, 2009: 272-277.

CUDA-based Parallel Visualization of 3D Data

XU Sai-hua, ZHANG Er-hua✉

Department of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract: Visualization in scientific computing is a new field of study proposed and developed in developed countries in the 20th century and the late 80's. Using Computer Graphics and Image processing techniques, 3D data can be converted to graphics or images displayed on screen for interactive processing. The main methods include surface visualization and volume rendering. Ray-casting algorithm is the most classical method of volume rendering methods about 3D data, which has higher image quality but longer calculation time. CUDA-based programming technique using the GPU hypercore parallel computing function, can significantly improve the calculation speed, and realize the real-time visualization of large-scale 3D data.

Key words: visualization; ray-casting; GPU; CUDA

作者简介：徐赛花（1986－），女，南京理工大学计算机科学与技术学院硕士研究生，主要从事科学计算可视化与虚拟现实技术等方面的研究，Tel：13951884315，E-mail：hhsherrill@163.com；张二华✉（1967－），男，1988 年毕业于中国地质大学（武汉）地球物理系，2000 年中国地质大学（武汉）获博士学位，南京理工大学计算机科学与技术学院副教授，主要从事三维数据场可视化、虚拟现实及模式识别方面的教学与研究，Tel：025-84315751-816，E-mail：zherhua@163.com。