基于CUDA的数字重建影像生成算法

杜晓刚 党建武 王阳萍
(兰州交通大学电子与信息工程学院 兰州 730070)

摘 要 鉴于数字重建影像生成过程具有良好的并行性,实现了一种基于CUDA并行计算的数字重建影像生成算法。该算法首先在CPU端使用八叉树结构来剔除体数据中的空体素并将其载入GPU;然后在GPU中根据光线和线程的对应关系,设计光线内核函数来模拟一束X线穿透人体组织的衰减过程;最后在GPU中由多线程并行执行内核函数来完成DRR图像生成过程。实验结果表明,该方法在保证DRR生成质量的前提下能有效利用GPU的并行计算能力,提高DRR图像的生成效率,满足图像引导放疗中对DRR生成过程的实时性要求。

关键词 数字重建影像,计算统一设备架构,图像引导放疗
中图法分类号 TP391.410 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.064

Generation Algorithm of Digital Reconstruction Radiographs Based on CUDA

DU Xiao-gang DANG Jian-wu WANG Yang-ping
(School of Electronic & Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract Because the generation procedure of digitally reconstructed radiograph has good parallelism, the digital reconstruction radiograph generation algorithm based on CUDA parallel computing was presented in this paper. Firstly, the octree structure is adopted to organize the volume data in CPU, and then the volume data are loaded into the GPU. The kernel function which can be used to simulate the decay process of X-rays penetrating the human body is designed according to the correspondence between the light and the thread, and finally the kernel function is executed in parallel by the multi-thread to complete the DRR image generation process. The experimental results show that this algorithm uses effectively the parallel computing capabilities of GPU in the premise of ensuring the quality of the DRR, significantly improves the generation speed of DRR, and meets the real-time requirements of DRR in the image-guided radiotherapy.

Keywords Digital reconstruction radiograph, Compute unified device architecture, Image guide radiotherapy

1 引言

数字重建影像(Digital Reconstruction Radiograph, DRR)是为了模拟X光片成像,对体数据进行光线投射并累加X光源穿透体数据后的衰减信息,最终在成像屏幕上产生的重建图像[1]。DRR图像已经被广泛地应用到介入手术和图像引导放疗等很多领域[2,3]。在重离子放疗计划系统中,为了制定合理的放疗计划,需要实时地从不同BEV方向显示DRR,另外在重离子放疗实施时,为了校正病人的摆位误差,需要用代表病人当前治疗位置的X线图像与利用CT图像生成的DRR图像进行配准,以保证治疗实施过程的精确性。因此针对大规模的CT体数据,快速实时地生成DRR图像显得非常重要。

根据传统光线投射算法的基本思想,虽然可以产生高质量的DRR图像[4],但在生成速度方面却不能满足图像引导放疗的实时性要求。为了提高算法效率,很多学者提出新算法来加速光线投射的过程。Wang等提出一种基于圆柱调和映射快速DRR技术[5],但该算法的绘制速度没有达到实时或交互绘制的要求;蔡文立等提出了一种结合Shear-warp和Splatting算法的混合加速方法,其在平行投影条件下能够将光线投射算法绘制速度提高到原先的5~6倍左右[6]。这些新算法比传统的光线投射算法取得了更快的计算速度,但依旧不能满足图像引导放疗中的实时性要求。W. Birkfellner等人提出了一种基于splat rendering的DRR图像生成方法并将其应用到2D-3D医学图像配准中,在PC机上针对大小为30MB左右的体数据,绘制能用于2D-3D配准的轻度模糊的DRR图像的时间为100ms[7]。该算法的DRR生成速度达到了实时性要求,但应用于放疗计划系统时其成像质量有待提高。

随着图形处理器(GPU)处理技术的快速发展,出现了很多基于GPU的光线投射算法。Stegmaier等提出了基于Shader Model 3.0的单步光线投射体绘制算法[8];梁承志等提出了基于Shader Model 4.0的单步空间跳跃加速光线投射算法[9],与传统方法相比,这些方法提高了生成速度,但仍受到固定的GPU流水管线的制约,不能充分发挥GPU的计算潜

到稿日期:2014-04-11 返修日期:2014-06-20 本文受国家自然科学基金项目(61162016),甘肃省科技支撑计划项目(1104FKCA102),兰州交通大学青年基金(2013005)资助。

杜晓刚(1985-),男,博士生,讲师,主要研究方向为医学图像处理;党建武(1963-),男,博士,教授,博士生导师,主要研究方向为智能信息处理;王阳萍(1973-),女,博士,教授,主要研究方向为医学图像处理。

能。伴随GPU硬件的进一步发展，NVIDIA公司在推出Gefore8800系列显卡的同时提出了统一计算设备架构(Compute Unified Device Architecture，CUDA)，开创性地改革了GPU的编程模式，显著地提升了GPU的并行计算能力[10]。Zhang提出了一种基于三次B样条和CUDA的光线投射体绘制算法[11]，实验结果表明该方法可以有效地实现医学图像三维可视化，但是该算法并没有应用于生成DRR图像。鉴于基于光线投射的DRR生成方法具有良好的并行性，本文基于CUDA并行处理环境实现了一种快速实时的DRR图像生成算法。该方法在保证DRR生成质量的前提下，有效地提高了DRR图像的生成速度，满足了图像引导放疗的实时性要求。

2 DRR生成算法

2.1 DRR成像原理

数字重建放射影像生成算法是以医学图像CT数据为输入，通过模拟X光线穿透人体不同组织器官时的衰减和曝光过程来获取类似X光片的成像技术。DRR图像的生成过程与医学图像CT体数据有密切的关系。通常情况下，CT值由X光线对组织的衰减与对水的衰减的比值表示，单位为H(Housfield)。其定义如式(1)所示：

$$CT=\frac{\mu-\mu_{water}}{\mu_{water}}\times 1000 \tag{1}$$

式中，μ为X光线在组织中的衰减系数，μ_{water}为X光线在水中的衰减系数。衰减系数μ可以由CT值计算：

$$\mu=\frac{CT\cdot\mu_{water}}{1000}+\mu_{water} \tag{2}$$

对式(2)进行进一步变换得到式(3)：

$$\mu=(CT/1000+1)\cdot\mu_{water}\cdot F \tag{3}$$

其中，F为转换因子。在实际中，单能CT束流为60keV，水的线性衰减系数为$\mu_{water}=0.206$。并且认为：CT值超过1000时为金属，取转换系数$F=0.812$；CT值在120到1000之间时为骨骼，取转换系数$F=0.306$；CT值低于120时为软组织，取转换系数$F=0.077$[12]。一束X射线会被射线路径中的物质不断吸收，吸收能量取决于该物质及射线的能量，射线束穿过组织后形成的衰减图形会记录在线探测器底片上。在人体中，不同的器官组织具有不同的密度，因而具有不同的线性衰减系数。X射线穿透物质的衰减模型如式(4)所示[13]：

$$I=I_0\times e^{-\int_0^L\mu(x)dx} \tag{4}$$

其中，I_0为X射线的初始强度，μ_i是组织i线性衰减系数，L是X射线的长度。式(4)的离散形式如式(5)所示：

$$I=I_0\cdot e^{-\sum\mu_i l_i} \tag{5}$$

其中，I_0和μ_i的意义与式(4)中相同，l_i表示射线穿过组织i的距离。式(5)为生成DRR图像的关键公式，其对应的X射线的衰减模型如图1所示。

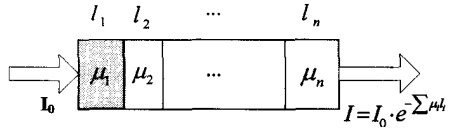


图1 X射线的衰减模型

2.2 算法思想

光线投射是一种经典体绘制方法，在计算机图形学和三维可视化方面得到了广泛的应用[14]。本文通过采用光线投射的原理来模拟X线穿透人体并经过人体组织吸收后衰减而产生DRR图像的过程。其主要过程是：从X线出束口发出多条X射线穿过三维体数据；在每条射线上进行等间距采样，并利用三线性插值方法由距离采样点最近的8个体素计算出该采样点的CT值对应的衰减系数；然后从前至后对所有采样点的衰减系数进行累加，得到该条射线对应成像平面的像素点的灰度值。针对每条射线重复上述过程后，将计算得到的所有像素点合成为一幅完整的DRR图像。基于光线投射原理的DRR图像生成过程如图2所示。

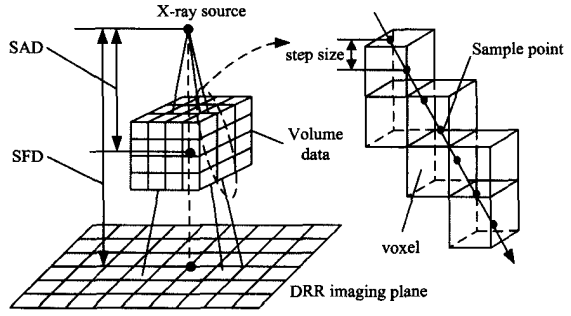


图2 基于光线投射的DRR图像生成过程

在图2中，SAD为X射线源点到体数据等中心的距离，SFD为X射线源点到DRR成像平面的距离。从X射线源点出发的射线条数与DRR图像的像素点个数相同，两者之间是一一对应的关系。为了简化采样计算过程从而提高采样效率，本文选择等步长和三线性插值来计算采样点的CT值。

3 CUDA编程模型

CUDA是一种不需要借助图形学API就能使用类C语言进行通用计算的开发环境和软件体系结构。CUDA编程模型主要包括网格(Grid)、线程块(Block)和线程(Thread)3层结构，每个Grid都由许多线程块构成，每个线程块由许多线程组成，其体系结构如图3所示[10]。线程块是CUDA应用程序执行的基本单位，且各线程块相对独立、互不影响，而同一个线程块内部的多个线程可以通过共享存储器来相互协作。线程块中用内建变量threadIdx来标识各线程，线程网格用内建变量blockIdx来标识各线程块[10]。

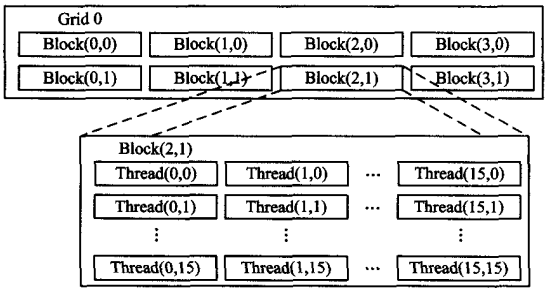


图3 CUDA的线程结构

CUDA应用程序需要由CPU和GPU协同工作来执行，其中CPU主要进行逻辑性强的串行计算和事务处理，而GPU主要负责执行高度并行的计算任务[10]。在执行CUDA应用程序时，CPU执行完串行代码后将数据传给GPU，CUDA应用程序的后续并行计算工作将交给GPU完成。CUDA应用程序的并行步骤被封装到内核函数中提供给GPU调用，当GPU调用内核函数时，GPU上所有线程同时并行执

行内核函数，各线程之间互不影响、完全并行。当所有线程都执行完内核函数后，GPU 再把计算结果传给 CPU 来完成后续处理和显示工作。

4 基于 CUDA 的数字重建影像算法

在基于光线投射的 DRR 生成算法中，由于从点光源投射出的所有光线之间互不影响，且每条射线穿透体数据进行采样并累加计算 CT 值的过程完全相同，因此其具有高度并行性。

4.1 八叉树结构优化体数据

通常情况下，三维体数据中的体素包括空体素和非空体素两种类型。空体素对最终的 DRR 图像生成没有贡献，只有对非空体素进行采样并合成才能得到有效的 DRR 图像。然而医学图像体数据中的空体素通常占到体素总数的 70%到 95%[15]，所以可以通过剔除体数据中的空体素来提高 DRR 图像的生成速度。

八叉树结构是由四叉树结构推广到三维空间而形成的一种规则的层次数据结构，在三维体数据的空间分解方面具有显著的优点，通过使用八叉树分解和遍历体数据，可以移除体数据中的空体素[16]。八叉树结构将三维体数据进行 8 等分，如果每一个部分内的物体属于同一属性就不再细分，否则将该部分再细分为 8 个部分；如此下去，直到每个体元内都属于同一属性为止。八叉树的每个节点有 8 个子节点或者没有子节点，这 8 个子节点是对父节点沿 X、Y、Z 3 个坐标轴的 2×2×2 规则细分。在八叉树结构中，每个维度增加一个细分层，这个维度的分辨率都将增大到原来的两倍，这也决定了八叉树中包含信息的叶节点深度。

本文算法首先采用八叉树来处理体数据，根据空间位置信息将体数据沿 X、Y、Z 3 个轴向进行分割，从而生成 8 个子块，如此逐层进行 8 方向分割，最终对体数据进行编码来去除其中的空体素，最后将处理后的体数据载入到 GPU 中进行并行处理。采用八叉树优化体数据的主要流程如图 4 所示。

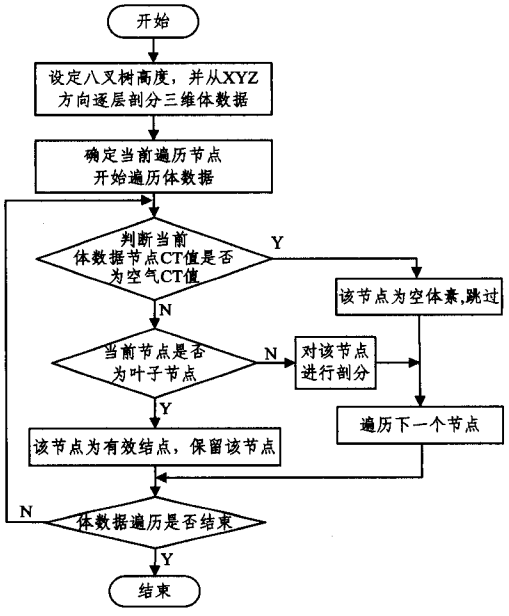


图 4 利用八叉树结构优化体数据的主要流程

4.2 线程分配

DRR 生成算法所需要的输入数据主要是由病人 CT 序列构成的体数据，基于 CUDA 实现 DRR 生成算法需要根据投射光线和 GPU 中线程的对应关系，考虑在 GPU 中进行线程分配。由于生成 DRR 的输入数据是医学 CT 图像序列，其分辨率通常为 512 * 512，因此从光源出发共有 512 * 512 条射线，每条射线对应 DRR 图像的一个像素，将一条射线穿透体数据进行采样并累加计算 CT 值的过程封装在内核函数中由一个线程完成，最终所有射线穿过体数据进行光线衰减产生二维的 DRR 图像。为了发挥 GPU 的并行计算能力，可将整个 DRR 图像屏幕作为一个网格，并将屏幕分为若干块，屏幕上每个像素点对应一个线程的执行结果，这样可保证整个并行计算过程充分利用 CUDA 的三层体系结构。因此根据上述分析分配线程如下：网格大小为 32 * 32，线程块大小为 16 * 16，每条光线在 X 和 Y 方向上都有 32 * 16＝512 条射线，每条光线路径上的采样点计算和 CT 值累加过程由一个线程独立执行内核函数完成。

4.3 CPU 端的数据处理

在本文算法中，CPU 作为主处理器主要为整个 DRR 生成过程的并行部分准备数据，并对并行部分的计算结果进行显示，在 CPU 上需要进行如下数据处理：

(1)将体数据读入内存并经过八叉树处理后，将体数据从内存拷贝到 GPU 中，并与 GPU 的纹理存储器绑定。

(2)由于常量存储器是只读存储器且读取速度快，将生成 DRR 时所需要的六自由度参数及 SAD、SFD 等参数从主机内存中读取到 GPU 的常量存储器中，从而提供给 GPU 中的各线程并行执行 kernel 函数时读取。

(3)在 CPU 的主存中建立 CT 值和物质衰减系数的关系查找表，并将该查找表载入到 GPU 的纹理存储器中用一维 CUDA 数组进行存储，以便各线程在获取采样点的 CT 值后，直接通过查表方式即可获取该采样点对应的衰减系数。

(4)使用 OpenGL 建立用于存储 DRR 图像像素数据的像素缓冲对象 PBO，以及建立窗口来显示 DRR 图像。通过调用函数 cudaGLMapBufferObject()将 PBO 映射到 CUDA 的地址空间，从而可以在 kernel 函数中来读写 PBO，随后将每个线程执行完 kernel 函数后的像素值保存在 PBO 中的对应位置，最后直接读取 PBO 来显示 DRR 图像。

4.4 GPU 端的数据处理

在 CPU 完成必要的数据准备工作后，在 GPU 上设计并执行 kernel 函数来完成光线投射及采样点合成。在设计和执行 kernel 函数时 GPU 端需要完成以下数据处理：

(1)计算视点和光线在空间中的位置。在 GPU 中执行 kernel 函数的所有线程都有对应的 blockID 和 threadID，通过这两个 ID 可以确定该线程在网格中的位置。每个线程在执行 kernel 函数时首先将线程 ID 转换为光线投射模型中定义的坐标，然后通过将初始坐标向量与变换矩阵相乘来计算视点坐标以及该线程对应的光线投射方向。

(2)根据视点坐标和该线程对应的光线投射方向来计算光线与体数据的交点。获取到光线和体数据的两个交点后，在该光线路径上由前向后进行等间距采样，然后使用纹理拾取函数获得采样点的 CT 值，且该 CT 值已经经过纹理缓存的三线性插值处理。再根据该 CT 值从 CT 值到衰减系数的查找表中获取到对应的衰减系数值，最后将所有采样点的衰减系数值进行累加，将累加结果转换为灰度值后保存到 PBO 对应的位置。当所有线程执行完 kernel 函数后，PBO 中即保存了 DRR 图像的灰度值。

4.5 本文的 DRR 生成算法流程

在本文的 DRR 生成算法中，首先将体数据读入到内存中并采用八叉树结构进行编码，然后将处理后的体数据和 GPU 并行计算所需要的参数都载入到 GPU 中，将光线穿透体数据的过程设计为内核函数在 GPU 上执行。本文 DRR 图像生成算法的基本流程如下：

Step 1　从医学图像 CT 序列中读取体数据，并采用八叉树结构对其进行编码处理，处理后将体数据载入到内存中。

Step 2　将体数据从内存中复制到 GPU 的纹理存储器中，为后续的 GPU 并行计算做准备。

Step 3　在内存中创建 CT 值和衰减系数的查找表，并将该查找表复制到 GPU 的纹理存储器中。

Step 4　在内存中设置六自由度参数、SAD 及 SFD 等参数，并将这些参数复制到 GPU 的常量存储器中。

Step 5　使用 OpenGL 创建 PBO 来存储 DRR 图像的像素数据，并创建窗口来准备显示 DRR 图像。

Step 6　确定网格大小和线程块的大小，并根据体数据的大小来计算线程和光线之间的对应关系。

Step 7　根据线程和光线的对应关系，GPU 中的每个线程需要并行地执行以下内核函数：

a)根据线程块和线程索引来计算对应的 DRR 成像平面像素点的位置；

b)计算 DRR 成像平面像素的物体空间坐标，并根据射线方向求出光线进入和离开体数据的交点坐标，确定光线位置；

c)沿着光线方向从光线和体数据的入射交点开始依次计算采样点位置坐标；

d)根据该采样点周围的 8 个原始体素点的 CT 值，利用三次线性插值算法计算得到采样点的 CT 值；

e)根据 CT 值与衰减系数的对应关系表获取该采样点的衰减系数值；

f)根据式(5)对所有采样点从前向后进行累加得到输出强度，再将输出强度转换成图像像素灰度值；

g)将图像像素灰度值写入 PBO 对象的对应位置。

Step 8　根据 PBO 数据来显示最终的 DRR 图像。

5　实验结果与分析

为了验证本文算法的正确性和有效性，我们选择了 6 组医学图像 CT 序列作为本文实验的测试数据，测试数据包括病人的头部和胸部的 CT 序列，其分辨率为 512 * 512，这些实验用的体数据的基本信息如表 1 所列。本文算法实现时采用 CUDA 4.0 版本的驱动库和运行时库，采用 Microsoft Visual Studio. NET 2008 作为编程环境，利用 CUDA C 作为编程语言。本实验的硬件平台配置如表 2 所列。

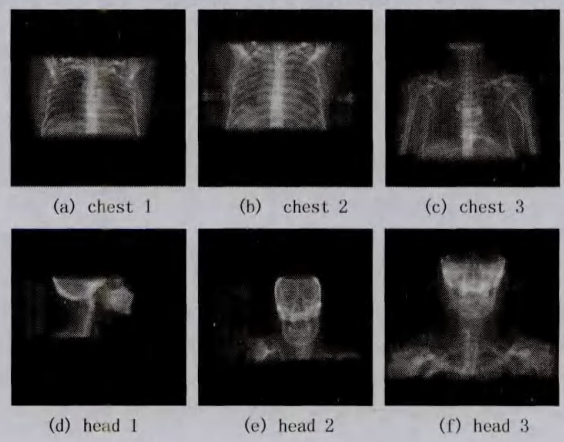
表 1　本文实验用的体数据基本信息

volume data	resolution	slices	size/MB
chest 1	512 * 512	40	20
chest 2	512 * 512	54	27
chest 3	512 * 512	66	33
head 1	512 * 512	30	15
head 2	512 * 512	40	20
head 3	512 * 512	50	25

表 2　实验环境配置

components	parameters
CPU	Inter core duo processor 2.0 * 2GHz
memory	2GB
The type of Video card	GeForce GTX650Ti
The memory of video card	1024MB
Multiprocessors * (192) CUDA cores	768 CUDA cores
Maximum number of threads per multi-processor	2048
Maximum number of threads per block	1024
Maximum sizes of each dimension of a block	1024 * 1024 * 64
Maximum sizes of each dimension of a grid	2147483647 * 65535 * 65535

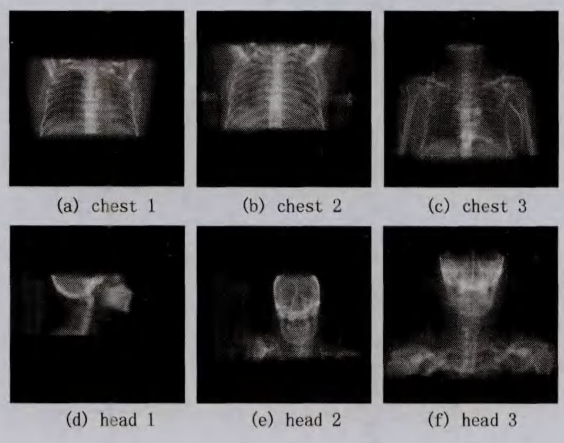
在本实验中，分别采用基于 CPU 实现的传统 DRR 生成算法和本文算法对该测试数据进行实验，生成 DRR 图像的分辨率均为 512 * 512，X 射线源点到等中心的距离 SAD 设置为 100cm，X 射线源点到 DRR 成像平面的距离 SFD 设置为 140cm，采用基于 CPU 的 DRR 成像算法生成的 DRR 图像如图 5 所示，采用本文算法生成的 DRR 图像如图 6 所示。通过比较图 5 和图 6 可以看出，使用基于 CPU 的传统 DRR 生成算法和使用本文算法所生成的 DRR 图像几乎没有差别，成像效果均比较好，成像质量都能满足图像引导放疗的需求。



(a) chest 1　(b) chest 2　(c) chest 3

(d) head 1　(e) head 2　(f) head 3

图 5　使用基于 CPU 的 DRR 图像生成算法得到的 DRR 图像

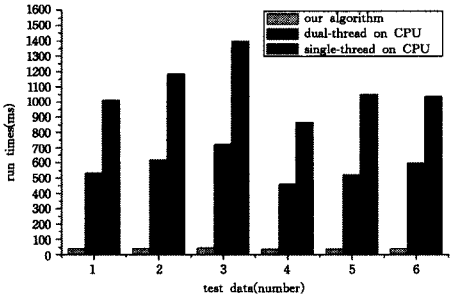


(a) chest 1　(b) chest 2　(c) chest 3

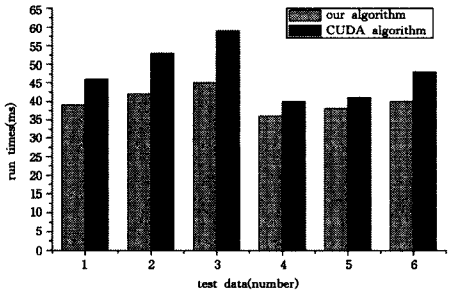
(d) head 1　(e) head 2　(f) head 3

图 6　使用本文算法得到的 DRR 图像

为了评价本文算法的计算性能，针对 6 组测试数据在相同的测试环境下对基于 CPU 实现的传统 DRR 生成算法(单线程和双线程)、基于 CUDA 的 DRR 生成算法及本文算法进行运行时间统计，结果如图 7 所示。



(a)本文算法与基于 CPU 实现的传统算法性能比较



(b)本文算法与基于 CUDA 的算法比较

图 7　3 种算法的性能比较

从图 7(a)中可以看出，针对 6 组测试数据集，使用本文算法生成一张 DRR 图像只需要 40ms 左右，而使用基于 CPU 单线程实现的传统 DRR 生成算法需要 1s 多，使用基于 CPU 两线程实现的传统 DRR 生成算法需要 0.5s 以上，本文算法的速度比基于 CPU 单线程实现的 DRR 生成算法快 30 倍左右，比基于 CPU 双线程实现的 DRR 生成算法快 15 倍左右。从图 7(b)中可以看出，针对 6 组实验数据，基于 CUDA 的 DRR 图像生成算法需要 50ms 左右完成 DRR 图像的生成，比本文算法慢 10ms 左右。这主要是由于本文算法采用八叉树结构对体数据进行了预处理，剔除了体数据中的大量空体素，从而提高了后续 DRR 生成过程的体数据访问速度，节约了算法执行时间。综上所述，本文算法对体数据采用八叉树进行预处理，有效地剔除了空体素，并在模拟光线投射过程中对投射光线采用 CUDA 架构进行并行计算，充分发挥了 GPU 的并行计算能力，提高了 DRR 的生成速度，能够满足图像引导放疗中校正病人摆位误差等过程的实时性要求。

结束语　针对图像引导放疗中对 DRR 的实时性要求，根据 DRR 生成过程的并行性，实现了一种基于 CUDA 的 DRR 生成算法。该算法首先在 CPU 端使用八叉树结构对体数据进行组织，然后将体数据载入 GPU，在 GPU 中对光线投射过程进行并行分解、多线程并行模拟 X 线穿透人体的衰减来完成 DRR 图像生成过程。实验结果表明，该方法在保证 DRR 生成质量的前提下有效地利用了 GPU 的并行计算能力，显著地提高了 DRR 图像的生成效率；并且该算法可以应用到 2D-3D 医学图像配准算法中来校正病人的摆位误差。同时由于 CUDA 是非常有效的并行计算体系结构，下一步工作将研究使用 CUDA 来加速重离子放疗计划系统中的其他模块，如重离子剂量计算过程，从而提高重离子放疗计划系统的运行效率。

参 考 文 献

[1] Kubias A, Deinzer F, Feldmann T, et al. 2D/3D image registration on the GPU[J]. Pattern Recognition and Image Analysis, 2008, 18(3): 381-389

[2] Khamene A, Bloch P. Automatic registration of portal images and volumetric CT for patient positioning in radiation therapy [J]. Medical Image Analysis, 2006, 10: 96-112

[3] 刘鹏，高军，雷勋祖，等. 一种基于光场的数字重建影像快速生成算法[J]. 南方医科大学学报，2007，27(10)：1537-1539

[4] Li Zhen-wei, Zhang Jian-guo. Study on volume rendering of CT slices based on ray casting[C]// The 3rd IEEE International Conference on Computer Science and Information Technology. 2010, 7: 157-160

[5] Wang F, Davis T E, Vemuri B C. Real-time DRR generation using cylindrical harmonics[C]// Proceedings of the 5th International Conference on Medical Image Computing and Computer Assisted Intervention. 2002: 671-678

[6] Cai W, Sakas G. DRR volume rendering using splatting in shear-warp context[C]// Proceedings of IEEE Nuclear Science Symposium and Medical Imaging Conference. 2001: 29-36

[7] Birkfellner W, Seemann R, Figl M, et al. Fast DRR generation for 2D/3D registration[C]// MICCAI. 2005: 960-967

[8] Stegmaier S, Strengert M, Klein T, et al. A simple and flexible volume rendering framework for graphics hardware-based ray casting[C]// Proceedings of Volume Graphics. 2005: 187-195

[9] Liang Cheng-zhi, Gao Xin-bo, Zou Hua, et al. Accelerated GPU ray-casting algorithm based on space leaping[J]. Journal of Image and Graphics, 2009, 14(8): 1684-1688

[10] NVIDIA CUDA. Compute unified device architecture: programming guide version 2.3[M]. Santa Clara, California, August 2009

[11] Zhang Chang-gong, Xi Ping, Zhang Chao-xin. CUDA-based volume ray-casting using cubic B-spline [C]// International Conference on Virtual Reality and Visualization. 2011: 84-88

[12] Hubbell J H. Photon Cross Sections, Attenuation coefficients and energy absorption coefficients from 10 KeV to 100 KeV [M]. NSRDS-NBS 29, 1969

[13] Dorgham O M, Laycock S D, Fisher M H. GPU accelerated generation of digitally reconstructed radiographs for 2D/3D image registration[J]. IEEE Transactions on Biomedical Engineering, 2012, 59(9): 2594-2603

[14] Parker S, Parker M, Livnat Y, et al. Interactive ray Tracing for volume visualization[J]. IEEE Transactions on Visualization and Computer Graphics, 1999, 5(3): 238-250

[15] 宋涛，欧宗瑛，王瑜，等. 八叉树编码体数据的快速体绘制算法[J]. 计算机辅助设计与图形学学报，2005，17(9)：1990-1996

[16] Laine S, Karras T. Efficient sparse voxel octrees [J]. IEEE Transactions on Visualization and Computer Graphics, 2011, 17(8): 1048-1059