GPU加速剂量计算中微分卷积/积分算法的实现

王先良[1,2] 刘乐乐[1] 吴章文[1] 勾成俊[1] 侯 氢[1]

1（四川大学原子核科学技术研究所 辐射物理及技术教育部重点实验室 成都 610064）

2（四川省肿瘤医院放疗科 成都 610041）

摘要 微分卷积/积分算法是计算精度较高的光子线剂量计算算法，较长的计算时间限制了该算法在临床上的使用。本文对微分卷积/积分算法中最耗时的部分实现了基于 GPU 的并行化计算，与基于 CPU 的计算相比，在 Tesla C1060 上计算速度提高可达 30X-60X。利用 γ 因子对计算结果的准确性进行了分析，结果显示，无论是均匀水模还是非均匀头模，在单照射野还是多照射野情况下，加速后的结果都与 CPU 的计算结果有相同的准确性。通过 GPU 并行加速，微分卷积/积分算法能成为日常的剂量计算算法。

关键词 CUDA，微分卷积/积分算法，GPU，剂量计算

中图分类号 TL72

剂量计算是放疗计划系统的核心内容之一，快速准确地提供感兴趣区内所受照射剂量的数据，对放疗计划的制定至关重要，如何在保证剂量计算精度的前提下，减少剂量计算时间是放疗领域研究的热点之一[1]。

提高剂量计算速度的方法主要有：采用不同的剂量计算算法和借助于计算能力更强的硬件。目前，放疗中研究和使用的剂量计算算法[2]基本分为蒙特卡洛算法、卷积/积分算法和有限笔形束算法。蒙特卡洛算法通常被作为剂量计算的标准，模拟了粒子与物质相互作用的全过程，能计算各种复杂条件下的剂量分布，但计算复杂，耗时长，临床实际应用较少，理论研究较多[3]。卷积/积分法又可分为三维卷积/积分[4]和二维卷积/积分[5]，采用空间不变的积分核计算的二维卷积/积分可以通过快速傅里叶变换加速剂量计算，所以也称为快速傅里叶法。有限笔束算法的核心思想是将照射野离散成许多相互独立的有限大小的射束元，介质中沉积在某点的剂量是各射束元在此点的剂量沉积的累加[6]。快速傅里叶法和有限笔束算法虽然有较快的计算速度，但都只能进行一维或二维的非均匀校正，不能准确模拟次级射线在非均匀介质中的分布，所以当介质非均匀梯度很大时，二维的非均匀校正可能带来较大剂量计算误差[7]。微分卷积/积分算法是一种基于点核的剂量计算算法，包括了光子和电子的各向散射，能进行三维非均匀修正，对不均匀组织及复杂结构的计算有较高的精度，是计算精度仅次于蒙特卡洛方法的光子线剂量计算算法[4,8]。另外该算法还能很好地进行非规则野和调强野的剂量计算，但较长计算时间限制了该算法在临床上的使用，为了使微分卷积/积分算法能成为日常的剂量计算算法，本文采用计算能力更强的硬件(GPU)对该算法进行加速。

GPU 最初是用于计算机图像显示和渲染的元件。现在的 GPU 不仅是功能强大的图像处理引擎，还可用于科学计算的高度并行化的可编程处理器。与传统的并行运算方式(如集群计算)相比，GPU 并行计算有很多优点：首先 GPU 有很多计算核心，硬件上适于做并行计算；其次，GPU 有较高的存储器读写带宽和浮点数运算能力，具备很高的计算性能；再次，基于 GPU 的并行运算所需投入少，能在个人电脑上实现，在低成本情况下就可获得不错的计算效率。近几年来，GPU 在可编程和硬件架构等方面的快速发展在各领域引发了大量关于 GPU 并行计算的研究，在科学计算方面扮演了越来越重要的角色[9]。

1 算法及GPU实现

1.1 微分卷积/积分算法

采用微分卷积/积分算法进行剂量计算需创建一个与射束中心轴方向垂直的虚拟体模(图 1(左))。图中虚线代表虚拟体模体元，实线代表初始体模体

国家科技支撑计划(2011BAI12B00)资助

第一作者：王先良，男，1986 年出生，2012 年 6 月于四川大学获硕士学位，从事医学物理研究

通讯作者：侯氢，qhou@scu.edu.cn

收稿日期：2012-11-22，修回日期：2012-12-20

元，虚拟体模体元的电子密度可以通过初始体模体元的电子密度插值获得，插值方法采用八点插值(图1(右))。图中的灰点代表虚拟体模体元，黑点代表初始体模体元。虚拟体模中处于位置$\vec{r}_v$的体元v在初始体模中的位置$\vec{r}_h$可以表示为：

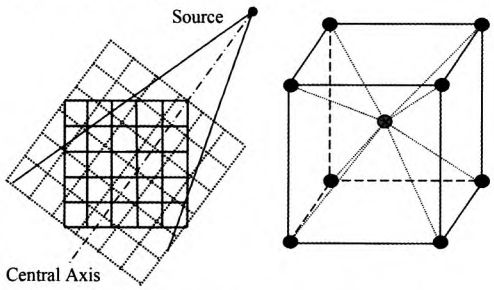

图 1 创建虚拟体模原理图
Fig.1 Schematic diagram of create virtual phantom.

$$\vec{r}_h = MAT(\theta_G, \theta_T, \theta_C)\vec{r}_v \tag{1}$$

其中，θ_G为机架角度，θ_T为治疗床角度，θ_C为准直器角度，MAT为射束坐标系到人体坐标系的转换矩阵。初始体模中，如果$\vec{r}_h$处于以其相邻八个体元的中心点为顶点的立方体里面，那么虚拟体模体元$\vec{r}_v$电子密度可以通过对$\vec{r}_h$周围八个体元的电子密度进行插值得到。

虚拟体模中，若能量为E的光子在$\vec{r}$处的比释总能T_E (total energy released per unit mass, TERMA)和剂量沉积核K_E已知，则任意剂量沉积点$\vec{r}^*$处的剂量沉积可由式(2)求出。

$$D(\vec{r}^*) = \int_E \int_V T_E(\vec{r}) K_E(\vec{r}^* - \vec{r}) \mathrm{d}V \mathrm{d}E \tag{2}$$

式中，T_E可用式(3)计算：

$$T_E(\vec{r}) = (r_0 / r)^2 \frac{\mu(E,\vec{r})}{\rho(\vec{r})} \rho(\vec{r}) \phi(\vec{r}_0) E \exp[-\int_{\vec{r}_0}^{\vec{r}} \mu(E,\vec{l}) \mathrm{d}\vec{l}] \tag{3}$$

式中，E为单能光子束能量，$\vec{r}_0$是入射点坐标，r_0是射线源到入射点的距离，r是射线源到作用点的距离，μ是能量为E的原射线在$\vec{r}$处的线性吸收系数，ρ是介质的密度，$\varphi(\vec{r}_0)$是入射点$\vec{r}_0$处的粒子注量强度分布，$\vec{l}$是积分路径。

式(2)中，剂量沉积核K是通过蒙特卡洛方法在水中模拟得到的，在计算非均匀介质时，需要对其做等效厚度修正。等效厚度的计算采用射线追踪，计算时以要计算的体元$\vec{r}^*$为起点，沿$\vec{r} - \vec{r}^*$方向进行射线追踪，通过式(4)计算体元$\vec{r}$和$\vec{r}^*$之间的等效厚度$d(x,y,z)$

$$d(x,y,z) = \sum \rho(x',y',z') l(x',y',z') \tag{4}$$

式中，$\rho(x',y',z')$是体元(x',y',z')相对于水的电子密度，$l(x',y',z')$是射线穿过体元(x',y',z')的几何长度(图 2)。假设体元$\vec{r}$和$\vec{r}^*$之间有一非均匀介质M，那么从体元$\vec{r}$发出的射线可认为是穿过了多层介质到达体元$\vec{r}^*$，在等效厚度修正下，体元$\vec{r}$对$\vec{r}^*$处剂量的贡献等效于水中体元$\vec{r}_e$对$\vec{r}^*$处剂量的贡献，即$K(\vec{r}^* - \vec{r}) = \tilde{K}(\vec{r}^* - \vec{r}_e)$，$K$是非均匀介质中的积分核，$\tilde{K}$是均匀介质中的积分核。

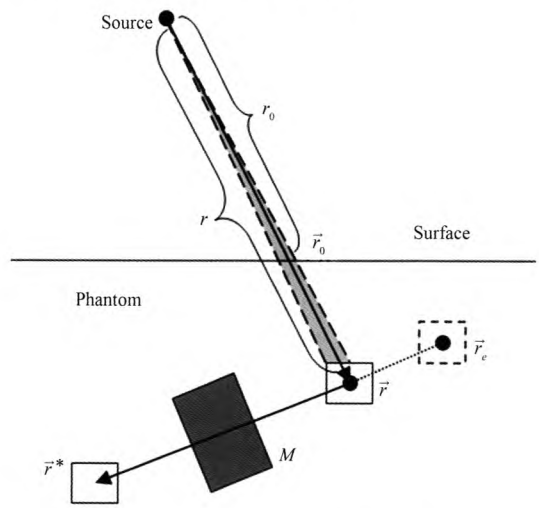

图 2 微分卷积/积分算法原理图
Fig.2 Schematic diagram of differential convolution/superposition algorithm.

对于多个照射野，由于不同角度的射束坐标系不同，为了方便后续进行数据处理，如等剂量线和DVH的显示，计算出的虚拟体模中的剂量还需再插值成初始体模中的剂量，这个过程与创建虚拟体模相似，剂量插值的过程仍采用八点插值。

1.2 GPU实现

本文采用CUDA(Compute Unified Device Architecture)架构实现GPU并行计算，CUDA是NVIDIA公司为GPU编程提出的一个全新的软硬件架构，它以C语言为基础，写出在GPU上执行的程序。CUDA是目前发展最成熟，应用最广泛的可编程GPU架构，详细的说明可以参见CUDA编程指南[10]以及相关文章中关于CUDA[1]的概述。

由§1.1可知，采用微分卷积/积分算法进行剂量计算大致分为四步：(1) 创建虚拟体模；(2) 计算TERMA；(3) 通过式(2)计算剂量；(4) 把虚拟体模中的剂量插值成初始体模剂量。剂量计算时，体模通常被离散成数百万甚至上千万个体元，照射野也被离散成成千上万条基本束，如果单纯依靠CPU计算，需要用循环语句依次遍历每个体元或基本束。与CPU不同，GPU采用并行线程替代循环语句，

一个 GPU 线程计算一个体元或基本束，利用 GPU 的多核，高浮点数运算性能加速剂量计算。

但 GPU 目前还不能完全取代 CPU，采用 GPU 进行大规模通用计算的合理模式是：用 CPU 控制主要流程并处理逻辑复杂的串行计算，把需要大量并行处理的计算放到 GPU 上处理。用 GPU 实现核函数主要有三个步骤：首先在 GPU 上为核函数所需数据分配内存空间，将 CPU 准备好的数据拷贝到 GPU 内存中；其次设定核函数的参数，在 GPU 上执行核函数，计算出结果；最后将 GPU 的计算结果拷贝到 CPU 内存中，解除对内存的绑定，释放 GPU 上已分配的内存空间。图 3 是用 CPU 和 GPU 完成微分卷积/积分算法的对比示意图。

在 GPU 实现核函数的过程中如何合理使用其内存是关键。CUDA 架构下的 GPU 内存分为：全局内存(global memory)、纹理内存(texture memory)、常量内存(constant memory)、共享内存(share memory)、寄存器内存(register memory)和局部内存(local memory)。不同的内存类型在分配方式和使用方法上都稍有区别[10]。全局内存没有或有极少的缓存，读取全局内存需要很长时间的延迟，对合并访问要求高，所以程序中尽量减少了全局内存的使用，对一些只读操作且占空间大的数组，如体元电子密度、积分核等，将其与纹理内存进行绑定，纹理内存有一定的缓存，并且在读取地址不连续的情况下能获得不错的数据读取速度。常量内存与纹理内存一样，是只读存储器，也有一定的缓存，但总空间较小，不能动态分配，所以只能将需要频繁访问的只读参数和已知固定大小的数组(如转换矩阵 *MAT* 等)放入常量内存。寄存器是高速缓存器，对寄存器的访问基本没有时间延迟，但是每个线程所用寄存器数目非常有限，如何控制寄存器的使用比较困难，因寄存器使用量不是完全根据核函数中定义变量的多少决定，线程中如果寄存器被用完或者定义了数组，数据就会存放在局部存储器中，对局部存储器的访问也需要很长的时间延迟，为获得好的加速效果，我们尽量减少了核函数中变量定义的个数。程序中恰当地使用这些内存能大幅提高计算速度，本文根据这些内存的特点对程序做了优化。

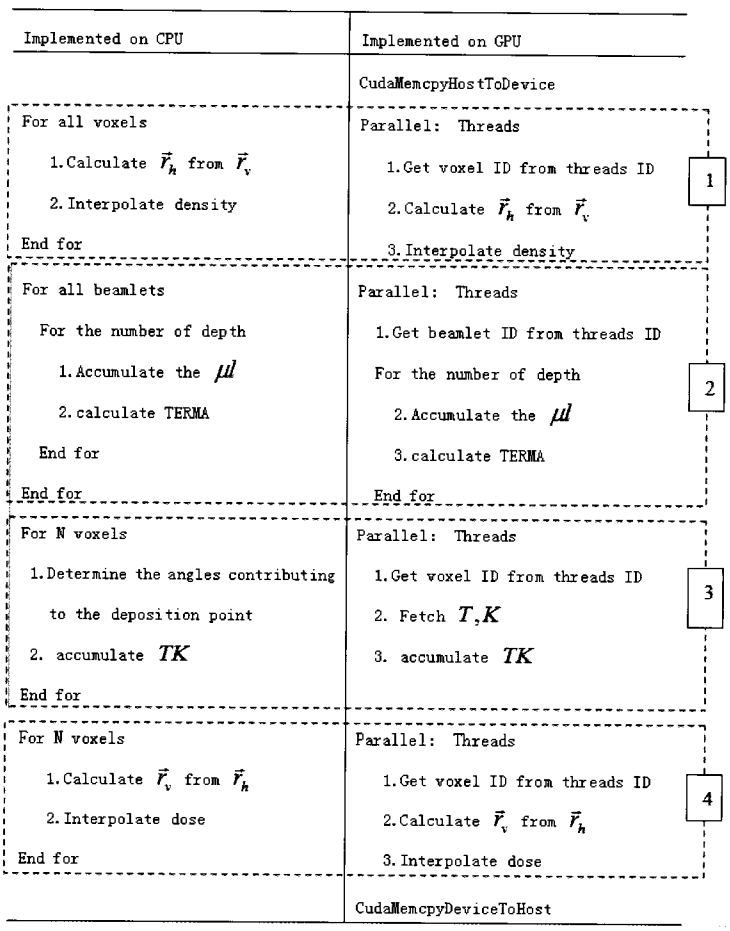

图 3 基于 CPU 和 GPU 的微分卷积/积分算法对比示意图

Fig.3 Comparison between CPU-based and GPU-based differential convolution/superposition algorithm.

2 结果与讨论

采用上述介绍的方法，我们编程实现了基于CUDA并行计算的微分卷积/积分算法，并对该算法的计算效率和计算精度进行了测试，测试平台如下：CPU为AMD Athlon四核630(主频2.8 GHz，内存4G)，显卡有NVIDIA Geforce GT320(72个CUDA核处理器，核心频率1302 MHz，显存1 G，显存位宽128 bit，显存带宽25.3 GB/S)和Tesla C1060(240个CUDA核处理器，核心频率1296 MHz，显存4 G，显存位宽512 bit，显存带宽102 GB/S)，操作系统为Win7，编程环境为Visual Studio 2008，CUDA版本是2.3。

测试选取了两个体模：一个是大小为30 cm×30 cm×30 cm的标准水模，体元大小为0.25 cm×0.25 cm×0.25 cm，另一个是由病人CT图像转化成的非均匀头模，大小为46.6 cm×46.6 cm×33.6 cm，体元大小为0.1 cm×0.1 cm×0.3 cm，虚拟体模体元的大小都为0.25 cm×0.25 cm×0.25 cm，野的大小为10 cm×10 cm，野的分辨率大小为0.5 cm×0.5 cm，计算一个照射野(0°)时6 MeV单能光子在体模内的剂量沉积。

为便于描述GPU加速的四个步骤的实验结果，用T_{CPU}表示程序在CPU上的执行时间，T^*_{GPU}表示核函数的执行时间，T_{GPU}表示程序在GPU上的执行时间(包括为数据在GPU上分配内存空间和将数据拷贝到GPU上所用的时间)，T_{CPU}/T_{GPU}表示加速倍数，Total是执行四个过程的总时间。

2.1 计算时间

表1给出了在计算水模和头模情况下，基于CPU与基于GPU所用时间的比较。由表1，通过GPU的并行运行机制，微分卷积/积分算法在低端显卡GT 320上获得了20X−30X的加速，在中高端显卡Tesla C1060上加速可以达30X−60X。

表1 水模和头模的计算时间与加速倍数

Table1 Execution time and speedup factors for water phantom and head phantom.

步骤 Step	GPU	T_{CPU} /s	T^*_{GPU} /s	T_{GPU} /s	T_{CPU} / T_{GPU}
水模 Water phantom					
1	GT320	0.608	0.016	0.171	3.56
	C1060	0.608	0.016	0.047	12.94
2	GT320	0.312	0.031	0.141	2.21
	C1060	0.312	0.015	0.109	2.86
3	GT320	1720.386	53.302	53.349	32.25
	C1060	1720.386	27.140	27.156	63.35
4	GT320	0.281	0.000	0.016	17.56
	C1060	0.281	0.000	0.016	17.56
总时间 Total	GT320	1723.387	—	56.128	30.70
	C1060	1723.387	—	28.953	59.52
头模 Head phantom					
1	GT320	0.952	0.016	0.265	3.60
	C1060	0.952	0.016	0.156	6.10
2	GT320	0.312	0.032	0.172	1.81
	C1060	0.312	0.016	0.140	2.23
3	GT320	1475.081	87.517	87.611	16.83
	C1060	1475.081	59.984	60.031	24.57
4	GT320	3.791	0.094	0.187	20.27
	C1060	3.791	0.031	0.093	40.76
总时间 Total	GT320	1485.861	—	93.789	15.84
	C1060	1485.861	—	64.375	23.08

由表1可见，无论是水模或头模，TERMA的加速倍数都偏低，造成这种现象的主要原因是计算量太小，两种体模采用CPU计算的时间都不到0.5s。GPU适合大数据量的并行计算，如果程序的计算量

太小，就无法隐藏内存访问以及数据传输的延迟，加速效果不明显。暂时对整个过程的加速倍数产生的影响不大，不过可以预见，随着对计算精度要求的不断提高，体元会划分得越来越小，射束细化后产生的基本束的数量会越来越多，加速效果也会越来越明显。

实验还研究了照射野大小对加速倍数的影响。通过对均匀体模的计算发现，加速倍数随照射野而增多(表 2)。造成这种现象的原因主要是用 CPU 计算球形积分时，有个函数可以判断对这个体元有影响的体元所占的空间范围，有此函数计算较小的照射野时可减少循环次数，计算速度较快。在 GPU 并行计算时有两个原因没有使用此函数。首先，此函数中有很多的逻辑运算语句，GPU 最初只用于图形计算，具有数据量大、数据相关性低、数据有相同的执行程序、并行度和计算密集性高等特点，所以在设计 GPU 硬件时，将大多数晶体管用于数据处理而不是数据缓存和逻辑控制。如果程序中逻辑运算语句多，GPU 并行计算的优势就不明显；其次，增加这个函数会增加很多变量，这些变量多数被存放在局部存储器中，GPU 对局部存储器的读取有很长的时钟周期延迟，太多的局部变量会减少加速倍数。测试结果也说明了这一点。通过对 2 cm×2 cm 照射野测试发现，GPU 中使用该函数比不用慢 40%。由此可见，在 GPU 并行计算中，直接将 CPU 代码变成 GPU 代码并不能获得很好的加速效果，还要根据实际情况对 GPU 代码做适当修改。

表 2　不同照射野的计算时间与加速倍数
Table 2　Execution time and speedup factors for different field sizes.

射野大小 Field size /cm^2	GPU	T_{CPU} /s	T_{GPU} /s	T_{CPU} / T_{GPU}
2×2	GT320	192.713	25.584	7.53
	C1060	192.713	17.078	11.28
5×5	GT320	566.229	33.868	16.72
	C1060	566.229	23.156	24.45
10×10	GT320	1723.387	56.128	30.70
	C1060	1723.387	28.953	59.52
15×15	GT320	3399.086	84.10	40.42
	C1060	3399.086	34.703	97.95

2.2　准确性

在提高计算速度的前提下，计算精度也是必须考虑的一个方面。由于所用的 CUDA 函数不是都满足 IEEE 754 标准[10]，所以需对 GPU 加速后的结果与 CPU 计算结果进行比较，放疗中剂量比较的方法有等剂量线比较法、剂量差比较法、等值间距法、γ 因子方法、规则化比较法、剂量体积直方图法等，本文采用 γ 因子方法[11]，距离标准选 3 mm，百分剂量差标准选 3%。

本文首先考虑了射野大小对 GPU 计算精度的影响。以 0°野为例，改变野的大小，发现照射野大小对水模计算精度没有影响，γ 通过率为 100%，对头模计算精度的影响很小，γ 值大于 1 的个数与体模体元总数之比低于 5×10^{-5}，可以忽略。

实验还考虑了射野个数对 GPU 计算结果的影响，通过对 1、5、9 个照射野(1 个野为 0°野，5 个野和 9 个野时是 360°均分野)的 CPU 和 GPU 结果对比发现，均匀水模 GPU 计算结果的 γ 通过率都是 100%，头模 γ 值大于 1 的个数与体模体元总数之比在 10^{-6}–10^{-5} 量级，说明加速后的结果满足精度要求。

3　结语

本文将微分卷积/积分算法程序与 GPU 相结合，用 CPU 控制主要流程，把能够大量并行处理的过程放到 GPU 上计算，与完全基于 CPU 的串行计算相比，通过把最耗时的部分实现 GPU 并行计算，该算法的计算速度在中高端 GPU(Tesla C1060)上可提高 30X–60X，GPU 加速后完成单个照射野的精确计算仅需 1 min 左右，在临床上是可接受的。通过 γ 因子验证，无论是单照射野还是多照射野情况下，加速后的结果都与 CPU 的计算结果有相同的准确性，可见本文实现的基于 GPU 的微分卷积/积分算法可满足临床应用要求。

参考文献

1　Pratx G, Xing L. GPU computing in medical physics: A review[J]. Medical Physics, 2011, **38**: 2685–2697

2　Ahnesjo A, Aspradakis M M. Dose calculations for external photon beams in radiotherapy[J]. Physics in Medicine and Biology, 1999, **44**(11): R99–155

3　Spezi E, Lewis G. An overview of Monte Carlo treatment planning for radiotherapy[J]. Radiation Protection Dosimetry, 2008, **131**(1): 123–129

4　Mohan R, Chui C, Lidofsky L. Differential pencil beam dose computation model for photons[J]. Medical Physics, 1986, **13**(1): 64–73

5　Mohan R, Chui C. Use of fast Fourier transforms in calculating dose distributions for irregularly shaped fields for three-dimensional treatment planning[J]. Medical Physics, 1987, **14**(1): 70–77

6 Bourland J D, Chaney E L. A finite-size pencil beam model for photon dose calculations in three dimensions[J]. Medical Physics, 1992, **19**(6): 1401−1412

7 Knoos T, Ahnesjo A, Nilsson P, *et al.* Limitations of a pencil beam approach to photon dose calculations in lung tissue[J]. Physics in Medicine and Biology, 1995, **40**: 1411

8 Vanderstraeten B, Reynaert N, Paelinck L, *et al.* Accuracy of patient dose calculation for lung IMRT: A comparison of Monte Carlo, convolution/superposition, and pencil beam computations[J]. Medical Physics, 2006, **33**: 3149−3158

9 Owens J D, Houston M, Luebke D, *et al.* GPU computing[J]. P Ieee, 2008, **96**(5): 879−899

10 Corporation N. NVIDIA CUDA Programming Guide. In: Corporation. N, ed. 3.0 ed2010: https://developer.nvidia.com

11 刘乐乐, 勾成俊, 吴章文, 等. 剂量分布比较中 γ 因子的快速计算方法[J]. 生物医学工程学杂志, 2012, **29**(3): 550−554
LIU Lele, GOU Chengjun, WU Zhangwen, *et al.* Fast γ index calculation method in dose distribution comparison[J]. Journal of Biomedical Engineering, 2012, **29**(3): 550−554

GPU-based ultra fast dose calculation using differential convolution/superposition algorithm

WANG Xianliang[1,2] LIU Lele[1] WU Zhangwen[1] GOU Chengjun[1] HOU Qing[1]

1 (*Key Laboratory of Radiation Physics and Technology, Institute of Nuclear Science and Technology, Sichuan University, Chengdu 610064, China*)

2 (*Department of Radiotherapy, Sichuan Tumor Hospital, Chengdu 610041, China*)

Abstract **Background:** Dose calculation plays a key role in treatment planning for radiotherapy, its performance and accuracy are crucial to the quality of treatment plans. Differential convolution/superposition algorithm is considered as an accurate algorithm for photon dose calculation; however, improvement on its computational efficiency is still desirable for such purpose as real time treatment planning. **Purpose:** The goal of this work is to boost the performance of differential convolution/superposition algorithm by devising a graphics processing unit (GPU) implementation so as to make the method practical for daily usage. **Methods:** In this work, we implemented a GPU-based version of the differential convolution/superposition algorithm, by which the most time-consuming parts are implemented on GPU. In order to fully utilize the GPU computing power, the algorithm is modified to match the GPU hardware architecture. **Results:** Compared with the algorithm completely running on CPU, the GPU-based algorithm can speed up 30-60 times on a Tesla C1060 with higher values corresponding to larger field size. Finally, we use γ index to analyze the accuracy of calculation results, no matter one field or multi-field, homogeneous phantom or inhomogeneous phantom, the GPU implementation has the same accuracy as the CPU implementation. **Conclusions:** GPU is a useful solution for satisfying the increasing demands on computation speed and accuracy of dose calculation. The GPU-based differential convolution/superposition can be feasible and cost-efficient for satisfying the increasing demands for either computation speed or accuracy by advanced radiation therapy technologies.

Key words CUDA, Differential convolution/superposition algorithm, GPU, Dose calculation

CLC TL72