Nuclear Science and Techniques **22** (2011) 349–352

Using GPU to calculate electron dose for hybrid pencil beam model

GOU Chengjun LI Xia HOU Qing WU Zhangwen*

Key Laboratory for Radiation Physics and Technology of Ministry of Education, Institute of Nuclear Science and Technology, Sichuan University, Chengdu 610064, China

Abstract Hybrid pencil beam model (HPBM) offers an efficient approach to calculate the three-dimension dose distribution from a clinical electron beam. Still, clinical radiation treatment activity desires faster treatment plan process. Our work presented the fast implementation of HPBM-based electron dose calculation using graphics processing unit (GPU). The HPBM algorithm was implemented in compute unified device architecture running on the GPU, and C running on the CPU, respectively. Several tests with various sizes of the field, beamlet and voxel were used to evaluate our implementation. On an NVIDIA GeForce GTX470 GPU card, we achieved speedup factors of 2.18 – 98.23 with acceptable accuracy, compared with the results from a Pentium E5500 2.80 GHz Dual-core CPU.

Key words GPU, HPBM, Dose calculation

1 Introduction

At present, most planning calculations for electron beam radiotherapy are based on pencil-beam model [1–6], which assumes that beam fields are composed of a finite number of pencil beams. Hybrid pencil-beam model (HPBM) is virtually taken from the Fermi–Eyges[7] multiple-scattering theory and the electron transport bipartition model (ETPM). For HPBM, the longitudinal part of the pencil-beam distribution function is determined by ETBM, which carefully considers inelastic scattering and bremsstrahlung of the incident electrons in the medium, while the transverse part is calculated by Fermi–Eyges theory with its parameters being modified by ETBM.

By embedding ETBM into Fermi–Eyges theory, the HPBM offers an efficient approach to calculate the three-dimension dose distribution from a clinical electron beam. E-beams are widely used in the radiation therapy, but a major difficulty in routine clinical procedure is to calculate the dose distribution accurately in a reasonable calculation time. High-speed dose calculation can be achieved with a CPU of high processing speed, a multi-core or many-core computational architecture, and a CPU-clustered supercomputer. But these solutions are neither readily available nor accessible to most clinical users, due to the cost to build or use a high-speed computation platform.

Stream processors (SPs) in general-purpose graphic processing units (GPGPUs)[8,9] has depicted an innovative scenario of handling massive floating-point computation and making high-performance computation affordable to general users. The NVIDIA Corp. has developed the Compute Unified Device Architecture (CUDA) programming model and software environment, and scalable parallel programs can be written using a straightforward extension of the C language. With graphic cards of its GeForce, GTX or Tesla series, GPU-based computing has been utilized to speed up computational tasks relevant to radiotherapy[10–14], such as CBCT reconstruction, deformable image registration and dose calculation. A speedup factor from tens to hundreds has been achieved.

In this paper, we report the development of a CUDA-based parallel computing framework for fast calculation of E-beam dose using HPBM. The HPBM is introduced briefly, the CUDA implementation of a HPBM framework is detailed, and performances of the

* Corresponding author. *E-mail address*: wuzhangwen@gmail.com

Received date: 2011-05-26

HPBM implementation on both GPU and CPU platforms are evaluated with a 30 cm×30 cm×30 cm water phantom irradiated by 20 MeV electron beam. Several scenarios with different sizes of the field, beamlet and voxel are tested to compare the implementation efficiency. By comparing the computation time and accuracy of GPU and CPU, the GPU implementation leads to a speedup factor from 2.18 to 98.23 with maximum relative discrepancy of 8.08×10^{-4}.

2 Methods and Materials

2.1 HPBM

According to HPBM, the dose distribution for an electron pencil beam at energy E can be written as[4,5]

$$D_h(x,y,z,E) = D_{\mathrm{bm}}(z,E)\frac{1}{\pi A_h(z,E)}\exp\left(-\frac{x^2+y^2}{A_h(z,E)}\right) \tag{1}$$

where, $D_{\mathrm{bm}}(z, E)$ is the depth dose of an infinite broad electron beam calculated by ETPM, $A_h(z, E)$ is the transverse scattering power of electron at depth z, which is calculated by HPBM[5]. For a rectangle field electron beam collimated at $z = 0$, the dose distribution can be calculated by Eq.(2),

$$\begin{aligned} D(x,y,z,E) &= \int_{-a}^{a}\mathrm{d}x'\int_{-b}^{b}\mathrm{d}y' D_h(x'-x,y'-y,z,E) \\ &= \frac{D_{\mathrm{bm}}(z,E)}{4}\left[erf\left(\frac{a+x}{\sqrt{A_h(z,E)}}\right)+erf\left(\frac{a-x}{\sqrt{A_h(z,E)}}\right)\right] \\ &\times\left[erf\left(\frac{b+x}{\sqrt{A_h(z,E)}}\right)+erf\left(\frac{b-x}{\sqrt{A_h(z,E)}}\right)\right] \end{aligned} \tag{2}$$

where, the rectangle electron beam is from $-a$ to a, and $-b$ to b along x- and y- directions, respectively.

The dose distribution determined by Eq.(2) is for an E-beam at single energy, while the dose distribution considering the energy spectrum involved in a clinical incident E-beam can be written as

$$D(x,y,z) = \int_0^{E_0} w(E)D(x,y,z,E)\mathrm{d}E \tag{3}$$

where, $w(E)$ is the energy spectrum weight of E-beam at an incident energy of E_0. Using the regularization method, we can unfold the energy spectrum from its dose measurement depth[16] with the following equations.

$$D_M(z,E_0) = \int_0^{E_{\max}} D_{bm}(z,E)w(E)\mathrm{d}E \tag{4}$$

The discrete form of Eq.(3) is

$$D(i,j,k) = \sum_{l=0}^{NE} w_l \sum_{m=LX}^{HX}\sum_{n=LY}^{HY} I_{m,n} D(x_{i,m}, y_{j,n}, z_k, E) \tag{5}$$

where, $I_{m,n}$ is an element of intensity matrix of the incident E-beam field, $x_{i,m}$ and $y_{j,n}$ are the distances between the voxel (i, j, k) and the beamlet (m, n) in x and y directions, respectively, and z_k is depth of the voxel (i, j, k).

When an E-beam irradiates on a homogeneous water phantom, every beamlet has the same dose distribution. Usually, a dose distribution kernel of the beamlet as Eq.(6) is pre-calculated to reduce calculation time.

$$D_b(i,j,k,E_0) = \sum_{l=0}^{NE} w_l D(x_i, y_j, z_k, E) \tag{6}$$

The total dose distribution can be calculated as,

$$D(i,j,k) = \sum_{m=LX}^{HX}\sum_{n=LY}^{HY} I_{m,n} D_b(i-m, j-n, z_k, E_0) \tag{7}$$

2.2 GPU implementation

GPU is designed specifically for highly data-parallel intensive computation applications. In November 2006, NVIDIA introduced CUDA, a general-purpose parallel computing architecture with a new parallel programming model and instruction set architecture. In this paper, CUDA with NVIDIA GPU (GeForce GTX470) card is used as the implementation platform.

In the CUDA environment, a GPU should be used in conjunction with a CPU. The CPU serves as the *host*, while the GPU is called the *device*. The main codes run on the *host*, invoking *kernels* that are executed in parallel on the *device* by using a large number of CUDA threads (NVIDIA 2010). A kernel can be executed N times in parallel on the GPU by N different CUDA threads. For convenience, CUDA threads are grouped to form thread blocks, and blocks are grouped to comprise grids. The numbers of blocks and threads should be explicitly defined when

invoking a kernel. The threads of a block are grouped into warps (32 threads per warp). A warp executes a single instruction at a time across all its threads.

In this paper, the dose calculation implemented as CUDA kernels runs on the GPU, while the remaining simple arithmetic operations run on the CPU. Each thread calculates the dose received by a set of voxels with the same (i, j) index.

Some global parameters, such as beamlet dose kernel, can be accessed by the threads simultaneously, which may lead to massive memory accesses so that the efficiency may be significantly impaired. This objection can be overcome by storing those parameters in texture memory as cached data.

2.3 Test cases

Electron beams from a 20 MeV clinical linac were used to evaluate calculation efficiency and accuracy. The energy spectrum and transverse scattering power parameters were unfolded from the measurement data[5]. The source skin distance (SSD) for each beam was 100 cm. Three beam field sizes were used: 4 cm×4 cm, 10 cm×10 cm, and 20 cm×20 cm, with the beamlet sizes being 0.1 cm×0.1 cm, 0.2 cm×0.2 cm and 0.25 cm×0.25 cm, respectively, and the voxel sizes being 0.1 cm×0.1 cm×0.1 cm, 0.2 cm× 0.2 cm×0.2 cm and 0.25 cm×0.25 cm×0.25 cm, respectively.

With the serial codes at CPU platform, the dose was calculated voxel by voxel. On the contrary, a large number of voxels could be calculated at the same time on the GPU platform. The depth doses of selected voxels with the same (i, j) index were calculated within a thread. The block size and thread size were variable in every case. They were equal to the numbers of voxels in the y- and x- directions, respectively.

3 Results and Discussion

The performances of the HPBM implementation on both GPU and CPU platforms were evaluated by a water phantom experiment[16]. A 30 cm×30 cm×30 cm water phantom was irradiated in each test.

The tests with different sizes of the field, beamlet and voxel were simulated to compare the implementation efficiency. The CPU codes were executed on a Pentium E5500 2.80 GHz Dual-core CPU, and the GPU codes ran on a NVIDIA GeForce GTX 470 card. The test conditions were listed in Table 1, where T_{CPU} is the sequential execution time with the CPU implementation and T_{GPU} is the execution time with the parallelized GPU implementation, including the data transfer time between CPU and GPU.

Table 1 Results of HPBM on CPU and GPU platforms for a water phantom with different sizes of the field, beamlet and voxel.

Beam field / cm^2	Beamlet size / cm^2	Voxel size / cm^3	Voxels	T_{CPU} / s	T_{GPU} / s	Speedup factor	Max. error / 10^{-4}
20×20	0.10^2	0.10^3	5 803 281	4360.34	44.39	98.23	0.98
	0.20^2	0.20^3	863 821	162.77	1.94	83.90	0.23
	0.25^2	0.25^3	499 849	60.18	1.11	54.22	0.00
10×10	0.10^2	0.10^3	1 713 481	375.08	4.91	76.39	8.08
	0.20^2	0.20^3	290 421	15.75	0.39	40.38	0.21
	0.25^2	0.25^3	182 329	6.20	0.23	26.96	0.00
4×4	0.10^2	0.10^3	421 201	18.94	0.72	26.31	4.56
	0.20^2	0.20^3	92 781	1.03	0.15	6.87	0.00
	0.25^2	0.25^3	67 081	0.61	0.28	2.18	0.00

The parallel computation was composed by CUDA kernels with different block sizes and thread sizes. The block size and thread size were equal to the voxel numbers in y and x directions, respectively, for each test.

The CPU computational time ranged from 0.61 to 4 360.34 s, depending on the beam field, beamlet and voxel sizes; while the GPU computation time on ranged from 0.28 to 44.39 s, i.e. a speedup factor of 2.18 to 98.23.

The most significant speedup factor appeared in the test with the largest beam field size and finest voxel and beamlet sizes. This means that GPU platform is more suitable to calculate massive data than CPU platform, as data size increases quickly with the field size, but with decreasing voxel and beamlet sizes. In addition, as the voxel number in *x*-direction is set as the thread size, the number of threads executing simultaneously decreases with the beam field size, but with increasing beamlet and voxel sizes.

The numerical accuracies of the GPU codes were listed in Table 1. The results show that there are some degree of discrepancy between CPU codes and GPU codes with fine beamlet and voxel sizes. The maximum relative error 8.08×10^{-4} appeared with the beamlet and voxel sizes of 0.1 cm×0.1 cm×0.1 cm and beam field size was 10 cm×10 cm. Such degree of discrepancy is acceptable in clinical applications.

4 Conclusions

In this paper we present the application of GPU in the calculation of clinical electron dose distribution with HPBM in a water phantom. Compare to that on the CPU platform, the calculation speed on the GPU platform can be improved significantly. The speedup factors vary with different beam field size and voxel and beamlet sizes. These results prompt that a faster electron beam treatment planning system could be expected when HPBM is used in clinical application on the GPU platform.

References

1 Boyd R A, Hogstrom K R, Rosen II. Med Phys, 1998, **25:** 2176–2185.
2 Jette D, Walker S. Med Phys, 1997, **24:** 383–400.
3 Boyd R A, Hogstrom K R, Starkschall G. Med Phys, 2001, **28:** 2096–2104.
4 Luo Zh M, Jette D, Walker S Z. Med Phys, 1998, **25:** 1954–1963.
5 Gou Ch J, Wu Z W, Luo Zh M. Med Phys, 2003, **30:** 415–423.
6 Hogstrom K R, Almond P R. Phys Med Biol, 2006, **51:** R455–R489.
7 Eyges L. Phys Rev, 1948, **74:** 1534–1535.
8 Hardwick J, Grand S L, Anderson J, *et al.* IEEE Micro, 2008, **28:** 13–27.
9 Nickolls J, Montrym J, Oberman S, *et al.* IEEE Micro, 2008, **28:** 39–55.
10 Jia X, Gu X, Sempau J, *et al.* Phys Med Biol, 2010, **55:** 3077–3086.
11 Zhou B, Cedric X Y, Danny Z, *et al.* Med Phys, 2010, **37:** 5593–5603.
12 Hissoiny S, Ozell B, Bouchard H, *et al.* Med Phys, 2011, **38:** 754–764.
13 Gu X J, Choi D, Men C, *et al.* Phys Med Biol, 2009, **54:** 6287–6297.
14 Men C, Gu X, Choi D, *et al.* Phys Med Biol, 2009, **54:** 6565–6573.
15 Luo Zh M, Jette D. Phys Med Biol, 1999, **44:** N177–182.
16 Shiu A S, Tung S, Hogstron K R, *et al.* Med Phys, 1992, **19:** 623–636.