GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues

Nunu Ren,[1] Jimin Liang,[1] Xiaochao Qu,[1] Jianfeng Li,[1] Bingjia Lu,[3] and Jie Tian[1,2]*

[1]*Life Sciences Research Center, School of Life Sciences and Technology, Xidian University, Xi'an 710071, China*
[2]*Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*
[3]*School of Electronic Engineering, Xidian University, Xi'an 710071, China*
**tian@ieee.org*

Abstract: As the most accurate model for simulating light propagation in heterogeneous tissues, Monte Carlo (MC) method has been widely used in the field of optical molecular imaging. However, MC method is time-consuming due to the calculations of a large number of photons propagation in tissues. The structural complexity of the heterogeneous tissues further increases the computational time. In this paper we present a parallel implementation for MC simulation of light propagation in heterogeneous tissues whose surfaces are constructed by different number of triangle meshes. On the basis of graphics processing units (GPU), the code is implemented with compute unified device architecture (CUDA) platform and optimized to reduce the access latency as much as possible by making full use of the *constant memory* and *texture memory* on GPU. We test the implementation in the homogeneous and heterogeneous mouse models with a NVIDIA GTX 260 card and a 2.40GHz Intel Xeon CPU. The experimental results demonstrate the feasibility and efficiency of the parallel MC simulation on GPU.

OCIS codes: (170.3660) Light propagation in tissues; (170.5280) Photon migration; (170.7050) Turbid media; (200.4960) Parallel processing.

References and links

1. R. Weissleder, and U. Mahmood, "Molecular imaging," Radiology **219**(2), 316–333 (2001).
2. R. Weissleder, and M. J. Pittet, "Imaging in the era of molecular oncology," Nature **452**(7187), 580–589 (2008).
3. B. W. Rice, M. D. Cable, and M. B. Nelson, "In vivo imaging of light-emitting probes," J. Biomed. Opt. **6**(4), 432–440 (2001).
4. V. Ntziachristos, J. Ripoll, L. V. Wang, and R. Weissleder, "Looking and listening to light: the evolution of whole-body photonic imaging," Nat. Biotechnol. **23**(3), 313–320 (2005).
5. G. Wang, Y. Li, and M. Jiang, "Uniqueness theorems in bioluminescence tomography," Med. Phys. **31**(8), 2289–2299 (2004).
6. A. H. Hielscher, "Optical tomographic imaging of small animals," Curr. Opin. Biotechnol. **16**(1), 79–88 (2005).
7. A. P. Gibson, J. C. Hebden, and S. R. Arridge, "Recent advances in diffuse optical imaging," Phys. Med. Biol. **50**(4), R1–R43 (2005).
8. B. C. Wilson, and G. Adam, "A Monte Carlo model for the absorption and flux distributions of light in tissue," Med. Phys. **10**(6), 824–830 (1983).
9. S. A. Prahl, M. Keijzer, S. L. Jacques, and A. J. Welch, "A Monte Carlo model of light propagation in tissue," Proc. SPIE IS **5**, 102–111 (1989).
10. L. V. Wang, S. L. Jacques, and L. Q. Zheng, "MCML—Monte Carlo modeling of light transport in multi-layered tissues," Comput. Meth. Prog. Biol. **47**(2), 131–146 (1995).
11. D. A. Boas, J. Culver, J. Stott, and A. Dunn, "Three dimensional Monte Carlo code for photon migration through complex heterogeneous media including the adult human head," Opt. Express **10**(3), 159–170 (2002).
12. H. Li, J. Tian, F. Zhu, W. X. Cong, L. V. Wang, E. A. Hoffman, and G. Wang, "A mouse optical simulation environment (MOSE) to investigate bioluminescent phenomena in the living mouse with the Monte Carlo method," Acad. Radiol. **11**(9), 1029–1038 (2004).
13. E. Margallo-Balbás, and P. J. French, "Shape based Monte Carlo code for light transport in complex heterogeneous Tissues," Opt. Express **15**(21), 14086–14098 (2007), http://www.opticsinfobase.org/abstract.cfm?URI=oe-15-21-14086.
14. L. V. Wang, and S. L. Jacques, "Hybrid model of Monte Carlo simulation and diffusion theory for light reflectance by turbid media," J. Opt. Soc. Am. A **10**(8), 1746–1752 (1993).

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

15. N. S. Zołek, A. Liebert, and R. Maniewski, "Optimization of the Monte Carlo code for modeling of photon migration in tissue," Comput. Meth. Prog. Biol. **84**(1), 50–57 (2006).
16. E. Alerstam, S. Andersson-Engels, and T. Svensson, "White Monte Carlo for time-resolved photon migration," J. Biomed. Opt. **13**(4), 041304 (2008).
17. E. Alerstam, T. Svensson, and S. Andersson-Engels, "Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration," J. Biomed. Opt. **13**(6), 060504 (2008).
18. Q. Fang, and D. A. Boas, "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units," Opt. Express **17**(22), 20178–20190 (2009), http://www.opticsinfobase.org/oe/abstract.cfm?URI=oe-17-22-20178.
19. NVIDIA CUDA Compute Unified Device Architecture - *Programming Guide*, Version 2.3 (2009).
20. N. Ren, and J. Tian, gpu-Molecular Optical Simulation Environment (2010). *http://www.mosetm.net*.
21. J. Arenberg, "Re: Ray/Triangle Intersection with Barycentric Coordinates," in Eric Haines, ed., Ray Tracing News, **1** (1988). *http://tog.acm.org/resources/RTNews/html/rtnews5b.html#art3*.
22. T. Möller, and B. Trumbore, "Fast, minimum storage ray-triangle intersection," J. Graphics Tools **2**, 21–28 (1997).
23. H. Lensch, and R. Strzodka, "Massively Parallel Computing with CUDA" (2008). *http://www.mpi-inf.mpg.de/~strzodka/lectures/ParCo08/*.
24. B. Dogdas, D. Stout, A. F. Chatziioannou, and R. M. Leahy, "Digimouse: a 3D whole body mouse atlas from CT and cryosection data," Phys. Med. Biol. **52**(3), 577–587 (2007).
25. G. Alexandrakis, F. R. Rannou, and A. F. Chatziioannou, "Tomographic bioluminescence imaging by use of a combined optical-PET (OPET) system: a computer simulation feasibility study," Phys. Med. Biol. **50**(17), 4225–4241 (2005).
26. S. Prahl, "Optical Properties Spectra" (Oregon Medical Laser Clinic, 2001). *http://omlc.ogi.edu/spectra/index.html*.

1. Introduction

As a rapidly developing biomedical research field, molecular imaging is defined as the *in vivo* characterization and measurement of biological processes at the cellular and molecular level [1,2]. Among molecular imaging, optical molecular imaging techniques, especially fluorescence molecular tomography (FMT) and bioluminescence tomography (BLT), have been attracting more and more attention due to the advantages of low cost, high sensitivity and nonionizing radiation [3–6]. The study of light propagation in biological tissue is the key problem of optical molecular imaging and many methods have been introduced [7].

Monte Carlo (MC) method is firstly introduced to study the light propagation in tissues by Wilson *et al.* [8]. Being a statistical method, MC method is then improved by many researchers to achieve a more accurate result. Prahl *et al.* introduced the anisotropy and internal light reflection into the light propagation [9]. For the light propagation in complex heterogeneous medium, Wang *et al.* proposed a MC modeling of light transport in multi-layered tissues (MCML) which has been widely used in the field of optical molecular imaging until now [10]. Boas *et al.* described a MC code 'tMCimg' based on a voxelized model [11]. Li *et al.* developed a MC simulation platform 'MOSE' based on a triangle mesh model [12]. The code 'TriMC3D' developed by Margallo-Balbás *et al.* is based on a triangle model combined with an octree organisation [13]. MC method has been generally considered as the gold standard of modeling light propagation in heterogeneous tissues and used to validate the results obtained by other models. But the main drawback of MC method is the extensive computational burden. Various efforts have been proposed to reduce the simulation time. Wang *et al.* proposed a hybrid model of MC simulation and diffusion theory [14]. Zolek *et al.* employed certain approximations to the calculations of logarithmic and trigonometric functions [15]. Alerstam *et al.* proposed a white MC model for time-resolved photon migration [16]. However, these acceleration techniques introduce diffuse approximation, the approximation of function, or the scaling transformation, which are all at the expense of the precision or the flexibility of the MC method.

Further more, MC simulation could be sped up by the parallel computation, such as CPU-clustered supercomputers. However, traditionally supercomputers are neither readily available nor accessible to most researchers and clinical users due to the prohibitively high cost of facility deployment and maintenance. Recently, a new parallel approach using general-purpose graphic processing units (GPUs) has been adopted to speed up the MC simulation. GPU is well suitable for the problems that can be parallel executed due to its special architecture. Compared to CPU clusters, GPU is easier to access and maintain because it is

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

much less expensive. Alerstam *et al.* have shown that GPU-based acceleration could be applied in MC simulation and obtained a massive speedup over traditional CPU code [17]. Fang et al. developed a parallel code named Monte Carlo eXtreme (MCX) based on 'tMCimg' [18]. But GPU parallel computation on triangle mesh model has not been emerged until now. Compared to the multi-layered model or the voxelized model, triangle meshes based geometry could realize a more flexible space structure and more precise boundary [12,13]. We report a GPU-based MC simulation of light propagation in complex heterogeneous tissues in this paper, and the tissue surfaces are constructed by triangle meshes recovered from MRI or micro CT data. On the basis of compute unified device architecture (CUDA) developed by NVIDIA [19], the parallel computation of MC simulation on GPU is realized and evaluated on both the homogeneous and heterogeneous models. The experimental results demonstrate the feasibility and efficiency of the proposed method.

This paper is organized as follows. The simulation of MC-based light propagation in heterogeneous tissues is first introduced in section 2.1. Section 2.2 briefly presents the CUDA based programming. In section 2.3, the parallel implementation of MC simulation based on CUDA is presented. The computational time of the MC simulation implemented on CPU and GPU is discussed in section 2.4. Section 3 presents the accuracy and the computational efficiency of the proposed method. Finally, conclusions are presented and the future work is summarized in section 4.

2. Method

2.1 Monte Carlo method

The MC method is based on randomly constructing a set of trajectories of photons propagation in tissues while the stepsize and direction of each trajectory depends on the absorption and scattering properties of tissues. Many researchers mentioned above have developed many programs to simulate light propagation in complex heterogeneous tissues. MCML described an external infinitely narrow beam and a multi-layered model. The program 'tMCimg' is based on an external pencil-beam and a voxelized model. However, the shapes of the internal light source and tissues in FMT or BLT are both arbitrary with limited size which cannot be simulated by the previous programs. To deal with these problems, Molecular Optical Simulation Environment (MOSE) based on MC method is developed by our group for the simulation of light propagation in complex heterogeneous tissues whose surfaces are constructed by triangle meshes [12].

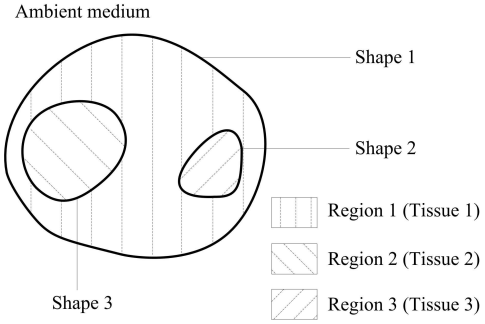

Fig. 1. Tissue structure in MOSE. Tissue 1 is the outermost tissue, Tissue 2 and 3 are both the internal tissues. Shape 1 is the boundary between Tissue 1 and ambient medium and corresponding to Tissue 1. Shape 2 is the boundary between Tissue 1 and Tissue 2 and corresponding to Tissue 2, so do Shape 3.

MOSE is a sequential version of MC simulation executed on CPU and the model described in MOSE is constructed by different numbers of tissues represented by independent optical properties and regions. A two-dimensional example of the tissue structure is shown in Fig. 1. For the convenience of description of the tissue structure and programming of the MC simulation, the tissues are separated into two different types: *internal tissue* and *outermost*

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

tissue. The tissue that has intersection with the ambient medium is defined as the *outermost tissue* and other tissues are defined as the *internal tissues* which are enclosed by the shape of the *outermost tissue*. The boundaries of the *internal tissues* are described by independent shapes without intersections and the regions of which are the parts included by the shapes, respectively. Correspondingly, the *outmost tissue* has the biggest shape, but the exact region of which is the part between the biggest shape and other shapes, and the biggest shape is the boundary between the *outmost tissue* and the ambient medium.

Except for the tissue structure, the process of light propagation in tissues based on MC method is similar to MCML, so the elaborate on all aspects of the process is skipped. Figure 2 shows the flowchart of the MC simulation which is slightly different from that in MCML. In this paper, the GPU-based parallel MC simulation named gpu-MOSE is based on the modification of MOSE [20].

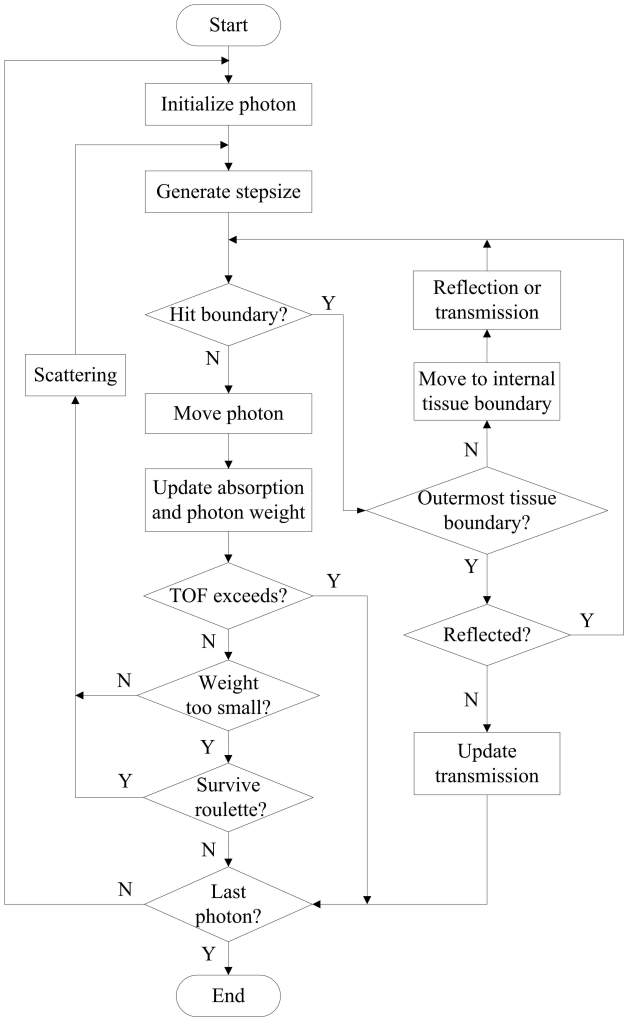

Fig. 2. Flowchart of photon propagation in tissues based on MC method.

As shown in Fig. 2, the photon is first generated from the light source. The weight of each photon is determined through dividing the total power of light source by the number of photon. The initial position and direction are generated according to the shape and the emergent angle of the light source. Once launched, the photon is repeatedly moved until it is terminated. There are three ways for a photon to terminate its propagation. First, it escapes the

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

outermost tissue to the ambient medium. Second, the photon may be absorbed by tissues while its weight drops below the predefined threshold. Third, the total time of flight (TOF) exceeds a predefined maximum value $T_{\max}$ which is set to 20ns in our work. During the propagation, the photon will transmit across the tissue boundary and we have to decide whether the photon enters into another *internal tissue, outmost tissue* or the ambient medium. If the photon is transmitted to the *internal tissue* or the *outmost tissue*, it will continue propagating with updated direction and stepsize. If the photon is transmitted to the ambient medium, the photon weight is scored into transmittance matrix depending on where the photon escapes.

2.2 CUDA-based GPU programming

The general-purpose computing on GPU (GPGPU) is well-suited for the data-parallel computational application in which the problem is divided and executed on a number of processor units in parallel. General graphic cards such as the GeForce series and the GTX series typically have 1-30 streaming multiprocessors (SMs) which are used to implement the parallel computation [19]. The multiprocessor employs an architecture called single-instruction, multiple-thread (SIMT) to manage hundreds of threads running several different programs. The CUDA platform allows the use of an extended C language to program the GPU.

In CUDA, the CPU is considered as the host while the GPU is called the device that operates as a coprocessor to the host. The threads executed on the device are grouped to thread block and the blocks are then organized into thread grid. The total number of threads is the number of blocks times the number of threads per block. CUDA assumes that both the host and the device maintain their own dynamic random access memory (DRAM), referred to as host memory and device memory, respectively. The device memory is an off-chip memory, mainly including *local*, *global*, *constant* and *texture* memory spaces, which is large but relatively slow (400-600 clock cycles of access latency). In addition, a GPU also contains a fast on-chip memory, including *registers*, *shared memory*, *constant cache* and *texture cache* spaces, which is small but much faster (below 4 clock cycles of access latency) than the off-chip memory. The read-only *constant cache* and *texture cache* can effectively speed up the access speed of the *constant memory* and *texture memory* spaces, respectively.

2.3 CUDA-based parallel MC simulation

In the parallel computation based on CUDA, the efficiency of the code is largely related to the memory management. Due to the limitation on the size and access speed of memory spaces within device, the data structures used in parallel MC simulation are specially designed to improve the computational efficiency.

In our work, the tissues are assigned different optical properties in reality, including the absorption coefficient μ_a, the scattering coefficient μ_s, the anisotropy coefficient g and the relative index of refraction n. These optical coefficients keep unchanged during the computation and the number of which is rather small so that they are stored in *constant memory*. The tissue shapes are constructed by large number of triangle meshes and are independent to each other. The vertex data of triangle meshes for all tissues will be frequently accessed during the parallel computation. Because the *constant memory* is too smaller to store the vertex data, the vertex data have to be stored in *texture memory*.

Because the tissue surfaces are constructed by triangle meshes, there are two important problems need to be considered during the MC simulation. One is to determine the relative position between the photon and the tissue surface constructed by triangle meshes, and the other is to compute the intersection between the photon path and the tissue surface. The two problems are both solved by a ray-triangle intersection evaluation [21,22]. A strategy of spatial partition is adopted to speed up the solving process as mentioned below.

In the strategy of spatial partition, a two-dimensional searching list for each tissue is firstly constructed. As shown in Fig. 3(a), the tissue surface is projected onto a 2D plane and

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

the projection of the triangle meshes is then partitioned by a uniform grid. Each grid cell is corresponding to an index list. The index of a triangle mesh is added to the index list if its projection overlaps the grid cell. The searching list for each tissue is constructed by assembling all the index lists. After all the searching lists for all tissues are constructed, the photon position and the path are also projected onto the 2D grid plane to determine the relative position and the intersection during the simulation, respectively [See Fig. 3(b)]. With the help of the partition strategy there would be small number of triangle meshes, belonging to the lists of grid cells (marked with red color in Fig. 3) intersected with the projection of the photon position or the photon path (marked with blue color in Fig. 3), need to be checked while conducting the ray-triangle intersection evaluation. Therefore, the time for determining the relative position of the photon or the intersection between the photon path and the tissue surface is reduced greatly.

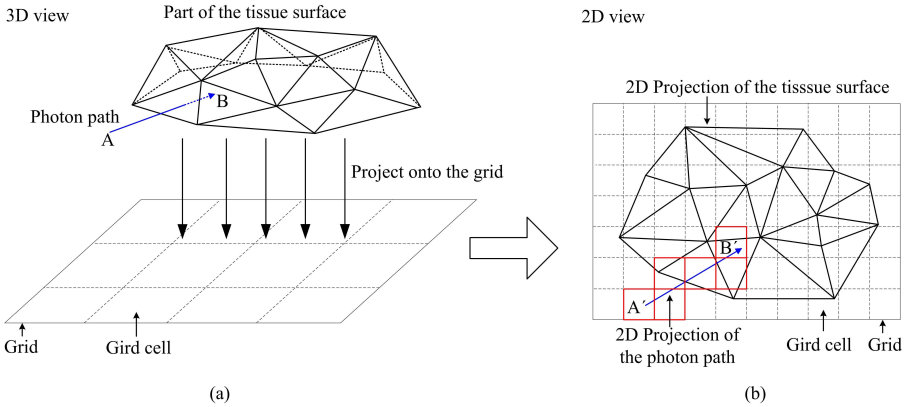

Fig. 3. (a) 3D view of the tissue surface which is projected onto the 2D plane grid. Points A and B are the starting and end points of a step (marked with blue color), respectively. A is the external point of the tissue and B is internal point of the tissue. (b) 2D view of the grid and the projections of the tissue and the photon path, A´ and B´ are the projections of points A and B, respectively. The grid cells intersected with the projection of the photon path are marked with red color.

Although the two-dimensional searching list can speed up the evaluation of ray-triangle intersection, the efficiency of the list executed on GPU is not as good as on CPU due to the discontiguous addressing of the list. Therefore, the index lists are merged into an one-dimensional searching array for each tissue, and all the searching arrays for all tissues are then merged into an one-dimensional total searching array. Although the array would keep constant during the simulation, the size of it is so big that it cannot be stored in *constant memory* as the optical properties of the tissues. In addition, access to it is related to the photon position during the MC simulation. For the position is random changed, it will be random-accessed accordingly. Therefore, *texture memory* is chosen to store the total searching array since it has enough memory space and suitable for the random-access.

Accessing *global memory* is an important part in GPU programming and this memory space is also used in parallel computation. Some data, including absorption and transmittance, need to be recorded and modified during the simulation process. Therefore, these data need to be recorded in the *global memory*. In our work we just record the transmittance for the comparison between CPU and GPU. During the simulation, different threads may access a same memory address while recording transmittance and the mutual interference will cause errors to the transmittance. This problem can be solved by the atomic operation which can perform the operation without interference from other threads, however, the atomic functions provided by CUDA cannot support the addition of single-precision floating-point until now. On the basis of the basic atomic functions, an atomic function for the float addition has been developed by Lensch and is adopted by us to record the transmittance [23]. The transmittance results on CPU and GPU are recorded and compared in section 3. Because the atomic

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

operation will prevent other threads to access the address while it is accessed by one thread, and the *global memory* is an off-chip memory with a large access latency, the efficiency of the parallel computation will be affected by the recordance of the transmittance.

Based on the above analysis, the flowchart of the parallel MC simulation and the specific usage of the memory spaces are shown in Fig. 4. In the parallel computation, the threads are executed in parallel to complete the MC simulation. During different stages of the parallel computation, the threads will access different memory spaces according to the type of accessed data. For example, the *texture memory* will be accessed for the vertex data or the total searching array, the *constant memory* will be accessed for the optical properties of the tissues, the *global memory* will be accessed for storing the transmittance results.

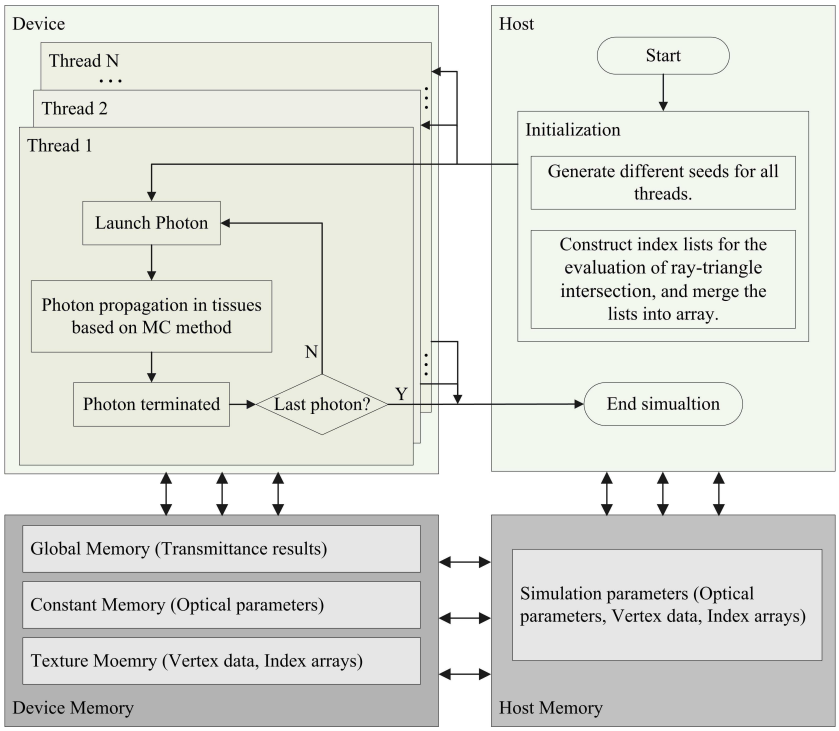

Fig. 4. Flowchart of the parallel MC simulation based on CUDA.

2.4 Computation time

Given the total number N_p of photons, the number N_b of *blocks* and the number N_t of *threads* per *block*. As shown in Fig. 5, the execution process and time cost of MC simulation on CPU and GPU are described by pseudocode, respectively. For the serial computation on CPU, the MC simulation is executed for N_p loops. For the parallel computation on GPU, the MC simulation on each thread is executed for $N_p/(N_b \times N_t)$ loops.

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

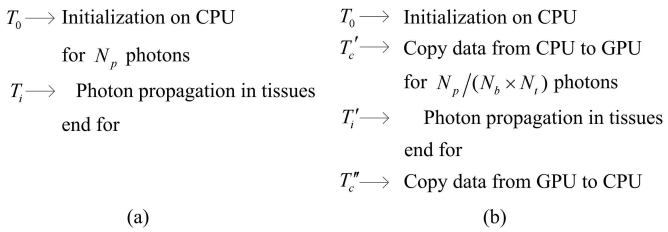

Fig. 5. Pseudocode description of MC simulation implemented on CPU and GPU, respectively.

The computational time for the MC simulation on CPU and GPU can be calculated respectively by

$$T_{cpu} = T_0 + T'_{cpu} = T_0 + \sum_{i=1}^{N_p} T_i \tag{1}$$

$$\begin{cases} T_j = \sum_{i=1}^{N_p/(N_b \times N_t)} T_i' \\ T_{gpu} = T_0 + T'_{gpu} = T_0 + (T_c' + \max(T_j) + T_c'') \end{cases} \quad j = 1, 2 \ldots N_b \times N_t \tag{2}$$

where T_0 is the time for the initialization executed on CPU before the MC simulation. T_c' and T_c'' are the computational time for copying data from host to device and from device to host, respectively. T_i and T_i' are the computational time of each photon propagation in tissues on CPU and GPU, respectively. T'_{cpu} is the total amount of time for simulating all photons on CPU. T_j is the computational time for each thread to run $N_p/(N_b \times N_t)$ photons on GPU. T'_{gpu} is the simulation time on GPU without the initialization time. T_{cpu} and T_{gpu} are the total computational time for the MC simulation on CPU and GPU, respectively. According to Eq. (2), the computational time on GPU depends on the maximum value among runtimes of all threads. Ideally, the total amount of time on GPU will decrease with the increase of the total number of threads. However, the linear decrease of computational time shown in Eq. (2) will be broken while the total number of threads is larger than a specific value determined by the hardware configuration of GPU. In addition, the time needed for transferring data between host and device is much bigger than that within devices. Thus, the communication time needs to be minimized to maximize the efficiency of parallel computation on GPU by moving more code from the host to the device, even if that means running kernel with low parallelism computations.

3. Results

According to the above analysis, gpu-MOSE based on CUDA is developed for speeding up the MC simulation. Various experiments are conducted to validate the accuracy and performance of gpu-MOSE and the results are illustrated as follows. Our work presented in this paper is implemented on a GTX 260 card which contains 24 multiprocessors, each of which contains 8 scalar processors (SPs).

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

3.1 Comparison to MCML

The program MOSE and gpu-MOSE are firstly verified through the comparison with MCML and two experiments are conducted. MCML has been widely used for the MC simulation of light propagation in tissues and has been recognized in the field of optical imaging. For the phantom of infinite slab in MCML is not supported in MOSE and gpu-MOSE. Therefore, the infinite slab is approximated by a flattened cylinder with 500mm of radius, centered at origin in MOSE and gpu-MOSE. The light source is an infinitely narrow pencil-beam with 1.0nW total power. The direction of the pencil-beam is parallel to the + Z axis and the origin is the incident point. The results about MCML shown below are obtained from [10].

The first experiment is based on a homogeneous slab with a matched boundary and the parameters are shown in Table 1. Ten MC simulations with each of 50000 photon packets are conducted to compute the averages and the standard errors (standard deviations of the averages) of the total reflectance and total transmittance, the results are shown in Table 2.

The second experiment is conducted in a semi-infinite scattering phantom with a mismatched boundary and the parameters are shown in Table 3. Ten MC simulations with each of 5000 photon packets are conducted to compute the averages and the standard errors of the total reflectance and total transmittance. The results are shown in Table 4. Since the phantom is a semi-infinite medium the transmittance is zero.

Table 1. Simulation parameters of the phantom

μ_a (1/mm)	μ_s (1/mm)	g	n	d (mm)
1	9	0.75	1	0.2

Table 2. The averages and the standard errors of the total reflectance and total transmittance

Program	Reflectance		Transmittance	
	Average	Standard error	Average	Standard error
MCML	0.09734	0.00035	0.66096	0.00020
MOSE	0.09746	0.00157	0.66067	0.00159
gpu-MOSE	0.09701	0.00082	0.65897	0.00094

Table 3. Simulation parameters of the phantom

μ_a (1/mm)	μ_s (1/mm)	g	n	d (mm)
1	9	0	1.5	semi-infinite medium

Table 4. The averages and the standard errors of the total reflectance

Program	Reflectance	
	Average	Standard error
MCML	0.25907	0.00170
MOSE	0.26110	0.00333
gpu-MOSE	0.26010	0.00406

As shown in Table 2 and 4, the results calculated by MOSE and gpu-MOSE agree with that calculated by MCML. Thus the MC simulation performed by MOSE or gpu-MOSE is feasible and reliable.

3.2 Performance evaluation

Three experiments with different models are performed to evaluate the performance of gpu-MOSE. The tissues are independently reconstructed from a volumetric mouse atlas provided by Dogdas [24]. The tissue surfaces are constructed by different number of triangle meshes and Fig. 6 shows the 3D surface rendering of these tissues. The tissue optical parameters summarized by Alexandrakis [25], as well as the absorption coefficients of oxy-haemoglobin (HbO2), deoxy-haemoglobin (Hb) and water reported by Prahl [26], are assigned as listed in Table 5.

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

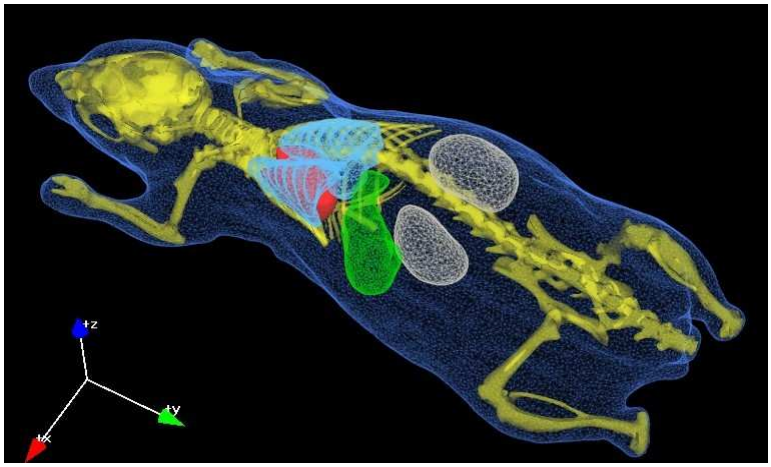

Fig. 6. 3D surface rendering of the tissues used in the experiments. The bounding box of the mouse phantom is 38 × 99.2 × 20.8 (mm). The point light source is located near the stomach marked with green color and its coordinate is (20, 50, 15) (mm). The picture is obtained from MOSE.

Table 5. Simulation parameters of the tissues

Tissue	μ_a (1/mm)	μ_s (1/mm)	g	n	Number of triangle meshes
adipose	0.005045	20.4545	0.94	1.35	2,000-100,000
skeleton	0.08138	26.0896	0.9	1.50	25,000
heart	0.078594	6.7104	0.85	1.42	3,500
lung	0.262956	36.818	0.94	1.38	6,000
kidney	0.088107	16.846	0.86	1.45	2,500
stomach	0.015044	18.4973	0.92	1.40	4,000

In the following experiments, an isotropic point source with 1.0nW total power is assumed as the light source and sampled by 10^6 photons. The CPU code of MC simulation is executed on a 2.40GHz Intel Xeon processor. Each data shown in Fig. 7 is the averaged value of three experiments.

In the first experiment, the relationship between the speedup and the number of threads ($N_b \times N_t$) is analyzed with different number of threads and the experimental result is shown in Fig. 7(a). The phantom is assumed to be homogeneous and is assigned the adipose optical properties. The number of the triangle meshes greatly influences the initialization time, so the phantom, whose surface is constructed by 5×10^4 triangle meshes, is used in this experiment to reduce this effect. Because the hardware resource (the number of SMs, stack, register and so on) on GPU is limited, the number of threads cannot be increased arbitrary so that the number of threads varied from 16 × 16 to 192 × 192 in this experiment. As shown in Fig. 7(a), the speedup becomes bigger at first and then keeps basically the same while the number of threads larger than 64 × 64. This result can be explained by the structural design of GPU, in which the performance of parallel computation on GPU is proportional to the utility rate of hardware resource on GPU, rather than the number of threads. At the same time, the utility rate of hardware resource is proportional to the number of threads at first, so the speedup increases correspondingly. However, the utility rate reaches the maximum approximately while the number of threads larger than a certain value which is 64 × 64 in this experiment, thus the speedup presents little change after the number of threads larger than 64 × 64. Through the experiment, we get the conclusion that the speedup is related to the utility rate of hardware resource and can be further increased through improving the utility rate of hardware resource on GPU.

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

For the purposes of the following experiments are irrelevant to the number of threads, the number of threads are set as 64 × 64.

The second experiment is conducted to study the relationship between the speedup and the number of triangle meshes. As shown in Fig. 7(b), the speedup is dramatically decreased with the increase of the number of triangle meshes. The reason is that the calculation cost and the number of access memory spaces on GPU are both proportional to the number of triangle meshes. The transmittance in MC simulation is recorded according to the position of photon escapes. In our work, the size of the transmittance matrix is equal to the number of the triangle meshes. Therefore, the address range of memory space and the time for storing the transmittance will vary with the number of the triangle meshes. The above analysis is supported by the results shown in Fig. 7(b).

In the third experiment, we perform the MC simulations based on heterogeneous models to analyze the influence of the branch and loop statements in the parallel computation. The number of the branch and loop statements is proportional to the number of tissues in the MC simulation. However, the parallel computation on GPU is not suitable to the branch and loop statements according to the structural design mentality of GPU. The simulations are performed with different number of tissues to demonstrate the above analysis. As shown in Fig. 7(c), time increased on GPU is more quickly than that on CPU while increasing the number of tissues. The speedup becomes smaller gradually and the experimental results demonstrate our deduction.

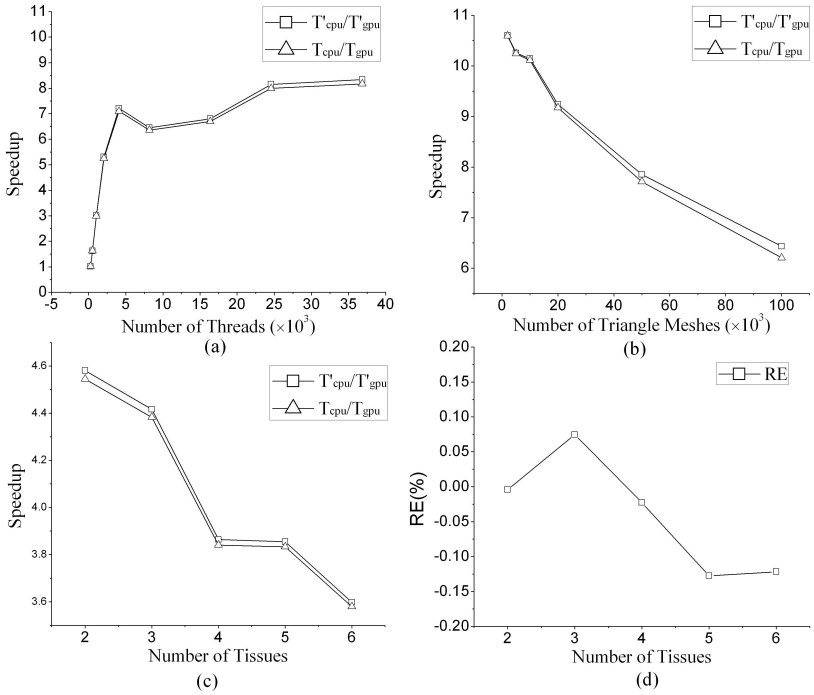

Fig. 7. (a) Speedup varies with the number of threads. (b) Speedup varies with the number of triangle meshes. (c) Speedup varies with the number of tissues. (d) Relative errors between the transmittances obtained from CPU and GPU.

From the comparisons between the speedup T'_{cpu}/T'_{gpu} and T_{cpu}/T_{gpu}, we can see that the initialization time also plays an important role. The initialization before parallel computation is necessary and takes some time, so the speedup T_{cpu}/T_{gpu} is always smaller than T'_{cpu}/T'_{gpu} which can be seen from the Fig. 7(a)–7(c). The second experiment [Fig. 7(b)] demonstrates

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

clearly that the initialization time is proportional to the number of triangle meshes. The speedup T_{cpu}/T_{gpu} is clearly affected by the initialization time while the number reaches 10^5.

The accuracy of the parallel computation is also considered in our work. The total transmitted power calculated through the transmittance matrix is compared in experiment 3. The absolute values of relative errors (RE) $(P_{gpu} - P_{cpu})/P_{cpu}$ are less than 0.15% as shown in Fig. 7(d), which show the reliability of the proposed GPU-based parallel MC simulation.

4. Discussion and conclusions

In this paper we develop a GPU-based parallel MC simulation for light propagation in complex heterogeneous tissues whose surfaces are constructed by different number of triangle meshes. The experimental results demonstrate the feasibility and efficiency of the GPU-based parallel computation. The parallel computation based on GPU has the advantage over that based on CPU due to the low cost and easy accessibility while achieve the same speedup. However, the speedup reported in other papers is approximately 10^2-10^3 times, which is much better than the value of about 10 times achieved in our work [17,18]. Through the experiments we get the reason is that the acceleration performance of GPU is affected by many factors, which are summarized as follows.

First, the acceleration performance is primarily related to the number of threads. Through the experimental results shown in Fig. 7(a), we can found that the speedup is non-linearly changed with the number of threads. In addition, the acceleration performance vary with different hardware configuration, including the number of multiprocessors, the memory size, the cache size, the clock rate and so on, even for the same number of threads. Generally speaking, the speed of the parallel computation is proportional to the level of the hardware configuration. The GPU card GTX 260 used in this paper has only 24 multiprocessors, 896Mb memory and 1.24 GHz. It can be predicted that a higher speedup factor would be obtained for a higher level of GPU card such as Tesla C1070 which has 4 GPUs, each with 30 multiprocessors. The usage of the atomic operation would reduce the parallel efficiency, but because of the few number of using atomic operation in our work, the time used for atomic operation accounted for a small proportion of the whole simulation time.

Second, the size of the data has a large influence on the speedup. The time needed for accessing data stored in different memory spaces is quite different. The memory spaces on GPU are assigned according to the date type and data size. There are a large number of data, including the triangle meshes and the simulation results, need to be stored in *texture memory* or *global memory* for the other memory space is too small. These data need to be used frequently during the simulation, which means that the *texture memory* or *global memory* needs to be accessed frequently. The time needed to access *texture memory* or *global memory* is much longer than that needed to access cache. Therefore, the extra time for accessing data on GPU is much longer than that on CPU with the increase of the data volume. The experimental results shown in Fig. 7(b) demonstrate this relationship. In our work, we have done our best to optimize the data store structure as far as possible for the purpose of reducing the number and the time cost of data access during the simulation.

Last, the complexity of the code for implementing the MC simulation reduces the efficiency of the parallel computation. The SIMT architecture at present is significantly affected by the flow control instruction (if, switch, do, for, while). In our program, the number of these statements is considerable and proportional to the number of the tissues, which can be analyzed from Fig. 1. The speedup would decrease with the increase of the number of tissues. This conclusion is demonstrated by the experimental results shown in Fig. 7(c). The efficiency of the parallel computation would be improved while reduce the number of the control instruction.

Outwardly the bottlenecks mentioned above all refer to the nature of GPU, the latter two factors are both relevant to the deficiency of the MC method proposed in this paper actually. Although the triangle mesh based model has the advantage of a more flexible space structure and more precise boundary over other models, the number of triangle meshes would be

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010

enormous for a good description of the model, and the triangle meshes are all different so that they have to be stored independently. Consequently, the huge amount of data is generated and will be accessed frequently during the MC simulation. In addition, the complex structure of the tissues used in our MC method leads to the complex program in which the number of control instruction is large and proportional to the number of tissues. Therefore, the method proposed in this paper doesn't make full use of the parallel computation on GPU.

Generally speaking, the parallel computation based on GPU is an excellent choice to reduce the computational time for implementing the MC simulation. With the fast development of GPU and the improvement of the programming of MC simulation, the bottlenecks mentioned above will be solved gradually and the speedup will have larger increase. In addition, the initialization in MC simulation can also be parallelized to further increase the speedup. We will continue to work on the GPU-based parallel computation in future. GPU-based parallel computation will play an important role in the field of optical imaging.

Acknowledgements

This work is supported by the Program of the National Basic Research and Development Program of China (973) under Grant No.2006CB705700, the Cheung Kong Scholars and Innovative Research Team in University (PCSIRT) under Grant No.IRT0645, the Chair Professors of Cheung Kong Scholars Program of Ministry of Education of China, CAS Hundred Talents Program, the National Natural Science Foundation of China under Grant No.30873462, 30900334, the CAS Scientific Research Equipment Develop Program (YZ0642, YZ200766), the Shaanxi Provincial Natural Science Foundation Research Project under Grant No. 2009JQ8018.

#123006 - $15.00 USD Received 19 Jan 2010; revised 15 Feb 2010; accepted 11 Mar 2010; published 17 Mar 2010