# GPU accelerated electric field Monte Carlo simulation of light propagation in turbid media using a finite-size beam model

**Yaru Wang,[1,2] Pengcheng Li,[1,2,*] Chao Jiang,[1,2] Jia Wang,[1,2] and Qingming Luo[1,2]**

[1]*Britton Chance Center for Biomedical Photonics, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Key Laboratory of Biomedical Photonics of Ministry of Education, Huazhong University of Science and Technology, Wuhan 430074, China*
*pengchengli@mail.hust.edu.cn

**Abstract:** An electric field Monte Carlo (EMC) simulation directly traces the complex electric field vectors in multiple scattering and estimates the electric field in a preferred direction. The full vectorial nature of EMC makes it a powerful and flexible tool to simulate the coherence and polarization phenomena of light. As a numerical method, EMC needs to launch a large number of photons to achieve an accurate result, making it time-consuming. Previously, EMC did not account for the beam size. Because of the stochastic character of the instantaneous electric field in the simulation, the convolution method alone is unsuitable for the Monte Carlo simulation of photon energy for a beam with a finite size. It is necessary to launch photons from all possible locations to simulate a finite-size beam, which results in a significant increase in the computational burden. In order to accelerate the simulation, a parallel implementation of the electric field Monte Carlo simulation based on the compute unified device architecture (CUDA) running on a graphics processing unit (GPU) is presented in this paper. Our program, which is optimized for Fermi architecture, is able to simulate the coherence phenomenon of a finite-size beam normally incident on turbid media. A maximum speedup of over 370x is achieved with a GTX480 GPU, compared with that obtained using an Intel i3-2120 CPU.



**OCIS codes:** (170.5280) Photon migration; (290.4210) Multiple scattering; (290.1350) Backscattering; (290.7050) Turbid media; (030.6140) Speckle.

## References and links

1. A. Ishimaru, *Wave Propagation and Scattering in Random Media* (Academic, New York, 1999), Vol.1.
2. J. W. Goodman, "Statistical properties of laser speckle patterns," in *Laser speckle and related phenomena*, 2nd ed., J. C. Dainty, ed. (Springer-Verlag, New York, 1984), pp. 9–75.
3. J. W. Goodman, *Speckle Phenomena in Optics: Theory and Applications* (Roberts and Company, Englewood, Colorado, 2007).
4. D. Huang, E. A. Swanson, C. P. Lin, J. S. Schuman, W. G. Stinson, W. Chang, M. R. Hee, T. Flotte, K. Gregory, C. A. Puliafito, and et, "Optical coherence tomography," Science **254**(5035), 1178–1181 (1991).
5. R. Drezek, A. Dunn, and R. Richards-Kortum, "Light scattering from cells: finite-difference time-domain simulations and goniometric measurements," Appl. Opt. **38**(16), 3651–3661 (1999).
6. L. Wang, S. L. Jacques, and L. Zheng, "MCML–Monte Carlo modeling of light transport in multi-layered tissues," Comput. Meth. Prog. Biomed. **47**(2), 131–146 (1995).
7. 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).
8. S. Bartel and A. H. Hielscher, "Monte Carlo simulations of the diffuse backscattering Mueller matrix for highly scattering media," Appl. Opt. **39**(10), 1580–1588 (2000).
9. M. Xu, "Electric field Monte Carlo simulation of polarized light propagation in turbid media," Opt. Express **12**(26), 6530–6539 (2004).

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

10. S. Moon, D. Kim, and E. Sim, "Monte Carlo study of coherent diffuse photon transport in a homogeneous turbid medium: a degree-of-coherence based approach," Appl. Opt. **47**(3), 336–345 (2008).
11. J. Sawicki, N. Kastor, and M. Xu, "Electric field Monte Carlo simulation of coherent backscattering of polarized light by a turbid medium containing Mie scatterers," Opt. Express **16**(8), 5728–5738 (2008).
12. M. Xu and R. R. Alfano, "Random walk of polarized light in turbid media," Phys. Rev. Lett. **95**(21), 213901 (2005).
13. M. Xu and R. R. Alfano, "Circular polarization memory of light," Phys. Rev. E Stat. Nonlin. Soft Matter Phys. **72**(6), 065601 (2005).
14. K. G. Phillips, M. Xu, S. Gayen, and R. Alfano, "Time-resolved ring structure of circularly polarized beams backscattered from forward scattering media," Opt. Express **13**(20), 7954–7969 (2005).
15. R. Liao, H. Zhu, Y. Huang, and J. Lv, "Monte Carlo modelling of OCT with finite-size-spot photon beam," Chin. Opt. Lett. **3**, S346–S347 (2005).
16. C. K. Hayakawa, V. Venugopalan, V. V. Krishnamachari, and E. O. Potma, "Amplitude and phase of tightly focused laser beams in turbid media," Phys. Rev. Lett. **103**(4), 043903 (2009).
17. M. Šormaz and P. Jenny, "Contrast improvement by selecting ballistic-photons using polarization gating," Opt. Express **18**(23), 23746–23755 (2010).
18. C. K. Hayakawa, E. O. Potma, and V. Venugopalan, "Electric field Monte Carlo simulations of focal field distributions produced by tightly focused laser beams in tissues," Biomed. Opt. Express **2**(2), 278–290 (2011).
19. J. Von Neumann, "Various techniques used in connection with random digits," J. Res. Natl. Bur. Stand. **5**, 36–38 (1951).
20. E. Alerstam, W. C. Y. Lo, T. D. Han, J. Rose, S. Andersson-Engels, and L. Lilge, "Next-generation acceleration and code optimization for light transport in turbid media using GPUs," Biomed. Opt. Express **1**(2), 658–675 (2010).
21. G. Marsaglia and A. Zaman, "A new class of random number generators," Ann. Appl. Probab. **1**(3), 462–480 (1991).
22. Nvidia Corporation, "CUDA programming guide 4.2," (2012).
23. G. Sendra, H. Rabal, M. Trivi, and R. Arizaga, "Numerical model for simulation of dynamic speckle reference patterns," Opt. Commun. **282**(18), 3693–3700 (2009).
24. D. D. Duncan and S. J. Kirkpatrick, "The copula: a tool for simulating speckle dynamics," J. Opt. Soc. Am. A **25**(1), 231–237 (2008).

## 1. Introduction

The coherence properties of light propagating in turbid media play an important role for optical radar object detection [1] and for coherent imaging applications such as laser speckle imaging [2,3] and optical coherence tomography (OCT) [4]. Generally, this phenomenon is described by Maxwell's equations, which can be solved by the perturbation theory or numerical approaches such as the finite-difference time domain (FDTD) method [5]. However, directly solving Maxwell's equations for a turbid medium, either analytically or numerically, is mathematically complex.

As is well known, light propagating through a random medium can be described by the radiative transfer equation (RTE), which neglects coherence properties and can be numerically solved by the Monte Carlo (MC) method [6–8]. Some improved MC methods [9,10], such as EMC, exist. By tracing the parallel and perpendicular components of the complex electric field, EMC is able to simulate the coherence properties of light, detect the backscattering speckle pattern and polarization state, and determine the Mueller matrix. The full vectorial nature of EMC makes it a powerful and flexible tool to simulate the coherence and polarization phenomena of light. Many applications of EMC have appeared in the literature. Coherence phenomena such as coherent backscattering (CBS), isotropization, randomization of helicity, and speckle patterns have been investigated [11–18].

However, as a numerical method, EMC needs to launch a large number of photons to achieve an accurate result, which makes it time-consuming. Meanwhile, the original form of EMC does not account for the size of the incident beam. Because of the stochastic character of the instantaneous electric field, the convolution method alone is unsuitable for the Monte Carlo simulation of photon energy for a beam with a finite size. It is necessary to launch photons from all possible locations to simulate a finite-size beam, which results in a significant increase in computation. For example, when using the same parameters as in Section 3.1, launching 100,000 photons of an infinitely narrow beam achieves a relative error of 0.078 (standard deviation divided by the mean) at the central point, whereas simulating a

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

Gaussian beam with waist radius $10\,l_s$ (the scattering mean free path) requires that more than 5,000,000 photons are launched to provide the same accuracy at the central point, which is 50 times the former. Moreover, at increased distances from the central point, the accuracy is lower. To solve this problem, we present a GPU accelerated EMC implementation under the compute unified device architecture, called CUDAEMC. With a GTX480 Fermi GPU, we achieve a 370x speedup when calculating backscattering speckle patterns with single precision compared to an i3-2120 CPU.

The detailed implementation of CUDAEMC is described in Section 2. Light propagating through an aqueous suspension of polystyrene spheres in slab geometry is simulated to validate the program in Section 3. In Section 4, the performance of CUDAEMC is analyzed, and the discussion and conclusion are presented in Section 5.

## 2. GPU-based EMC

A GPU device was used to accelerate the computing speed of EMC in parallel. All the functions of Xu's EMC [9] are realized. The original EMC program modeled an infinitely narrow beam. We expand the beam model to include finite-size beams, such as a Gaussian beam, and circular and rectangular flat beams. In this section, we present theory of EMC, model of the light beam, GPU implementation in detail.

### *2.1 Theory of EMC*

A detailed introduction to EMC can be found in the literature [9]. The medium in which light propagates is considered to be an aqueous solution of polystyrene spheres inside a slab. The light beam is divided into a large number of independent photon packets. Each photon packet possesses the parallel and perpendicular components of the complex electric field, which are tracked as the photons travel and scatter in the turbid medium. The scattering mean free path $l_s = 1/\mu_s$ is used as the length unit, where $\mu_s$ is the scattering coefficient. The three-dimensional Cartesian coordinate system is used. The plane of incidence, on which light is incident, is z = 0, and the positive direction of the z-axis points into the medium.

The Mie theory is used to describe scattering events, and the electric fields after the $n^{\text{th}}$ scattering $E_{\parallel n}$ , $E_{\perp n}$ are updated according to

$$\begin{pmatrix} E_{\parallel n} \\ E_{\perp n} \end{pmatrix} = L(\theta,\varphi) \begin{pmatrix} E_{\parallel n-1} \\ E_{\perp n-1} \end{pmatrix} \tag{1}$$

with

$$L(\theta,\varphi) = \left[F(\theta,\varphi)\right]^{-1/2} \begin{pmatrix} S_2\cos\varphi & S_2\sin\varphi \\ -S_1\sin\varphi & S_1\cos\varphi \end{pmatrix} \tag{2}$$

where $F(\theta,\varphi)$ is the light intensity normalization factor, given by

$$\begin{aligned} F(\theta,\varphi) = {} & (|S_2|^2\cos^2\varphi + |S_1|^2\sin^2\varphi)\,|E_{\parallel}|^2 + (|S_2|^2\sin^2\varphi + |S_1|^2\cos^2\varphi)\,|E_{\perp}|^2 \\ & + 2(|S_2|^2 - |S_1|^2)\cos\varphi\sin\varphi\,\mathrm{Re}[E_{\parallel}(E_{\perp})^*] \end{aligned} \tag{3}$$

The joint probability density function $p(\theta,\varphi)$ of the scattering angle $\theta$ and the azimuthal angle $\varphi$ is then

$$p(\theta,\varphi) = \frac{F(\theta.\varphi)}{\pi x^2 Q_{sca}} \tag{4}$$

where $Q_{sca}$ is the scattering efficiency.

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

The probability density function of $\theta$ can be calculated as

$$p(\theta)=\int_0^{2\pi} p(\theta,\varphi)d\varphi=\frac{|S_1(\theta)|^2+|S_2(\theta)|^2}{x^2Q_{sca}} \tag{5}$$

where $x$ is the size parameter and $x=2\pi na/\lambda$; $a$ is the particle radius.

Then, $\varphi$ is determined by the following formula:

$$p(\varphi\,|\,\theta)=\frac{p(\theta,\varphi)}{p(\theta)} \tag{6}$$

The optical path length $l$ and photon weight $w$ are also recorded. The electric field $E_d$ at the $n^{\text{th}}$ scattering in a given direction is determined as $E_d=(E_{\|n}x_n+E_{\perp n}y_n)\exp(2\pi il/\lambda)$, where $x_n$ and $y_n$ are the local coordinates after the $n^{\text{th}}$ scattering.

*2.2 Model of the light beam*

In EMC, all photon packets are launched normally to the slab from the location (0, 0, 0). When illuminated by a finite-size beam, the x- and y-coordinate values have a limited range. In CUDAEMC, the x- and y-coordinate values are random numbers rather than fixed values. We control the spatial distribution of the location of photons launched, so that beams with a particular distribution can be formed. Here we provide Gaussian beam, circular and rectangular flat beam models. The flat beam is composed of photon packets following a uniform distribution in a rectangular or circular area. For the Gaussian beam, we assume that the waist of the beam is in the plane of incidence. Thus, the locations from which photons are launched are random coordinates, which follow a two-dimensional Gaussian distribution. The Gaussian distribution is generated by the Box Mueller algorithm.

*2.3 GPU implementation*

In contrast with a CPU, a GPU has tens to hundreds of stream processors (SPs). Each SP can execute an independent thread. When running EMC on a GPU, the computations are divided into a number of blocks, each of which contains a number of threads. During simulation, each thread traces the path and scattering events of one photon packet independently. However, as memory accesses here are random, computation without optimization will lead to a low efficiency. Thus, coalesced access is introduced, and there is a corresponding change of calculation steps in each thread.

2.3.1 Calculation steps

The detailed calculation steps are shown in Fig. 1. When the main program starts in the CPU, tables of S1 and S2 (the nonzero components of the scattering matrix) for a spherical particle are calculated by the Mie scattering theory, and are used to calculate the scattering of photons later in the GPU. Another mapping table for $\theta$ is also produced according to Eq. (5). Each time a GPU kernel is started it returns to the CPU after a fixed number of GPU loops, and then, the CPU checks whether calculations for all the photons have been completed. If not, a new CPU loop starts, otherwise the results of the simulation are saved. Both the calculation of the movement and scattering of photons, and the estimation of electric field on the surface of the slab media are executed in parallel in the GPU. As photon parameters calculated in different stream processors are independent, the corresponding global memory locations that need to be accessed are stochastic, reducing the accessing speed. Several optimizations were performed here, as it was found that the access speed of global memory was the bottleneck for CUDAEMC. Herein, we realize the coalescing of the global memory accesses to some extent by temporarily storing the electric fields and Stokes vectors in shared memory, as described in detail in Section 2.3.4. Note that even when photons exit from the slab media, associated

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

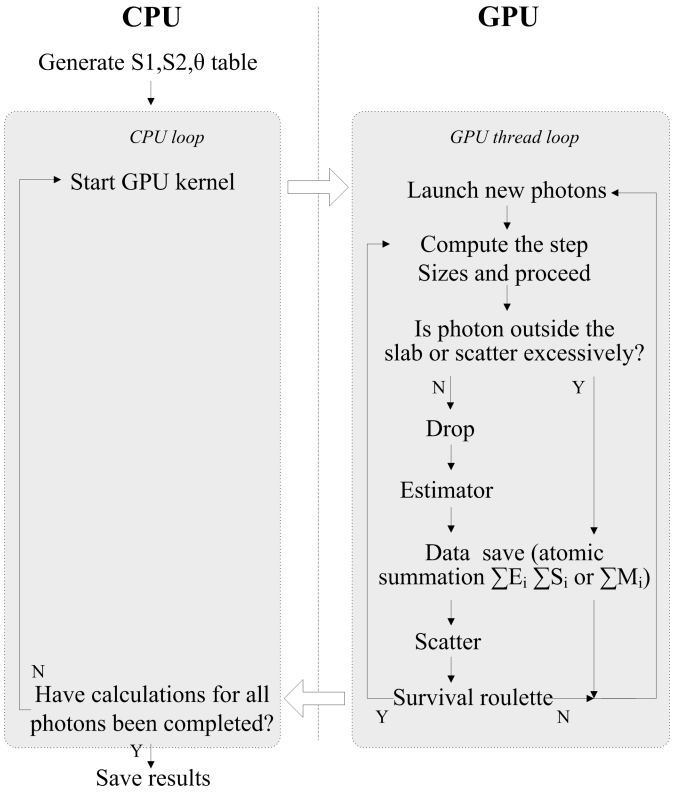


Fig. 1. Block diagram of the simulation process.

threads in the SP are not idle and help to save data to global memory. The specific steps in the GPU are listed as follows:

1. Launch new photon packet with initialized photon direction, position, weight and parallel and perpendicular components of electric field.
2. Generate a random step size following a negative exponential distribution with mean length $l_s$ and move the photon.
3. If the photon is outside the slab or has been scattered more than a predetermined number of times, skip to Step 5, otherwise proceed to Step 4. The reason for this step is explained in detail in Section 2.3.4.
4. Calculate the absorption, and then estimate the electric field in a preferred direction. Calculate the Stokes vector according to the electric field.
5. All threads (even those for photons outside the medium) add the electric field and Stokes vector or Mueller matrix using atomic functions. The electric field adds coherently yielding the instantaneous field $E_\alpha$ (α represents x, y or z, the instantaneous light intensity can then be determined as $I_\alpha = |E_\alpha|^2$). The Stokes vector adds incoherently and the average intensities are determined as $\langle I_x \rangle = (\langle S_1 \rangle + \langle S_2 \rangle)/2$, and $\langle I_y \rangle = (\langle S_1 \rangle - \langle S_2 \rangle)/2$. The $I_\alpha$ (with probability density function $p(I_\alpha) = \exp(-I_\alpha/\langle I_\alpha \rangle)/\langle I_\alpha \rangle$) follows the exponential distribution

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

$\exp(-\eta)$ . A judgment is made based on the locations and numbers of scattering events of the photons loaded in all the threads. If the photon exits the slab or the number of scattering events is larger than a defined number, go to Step 1.

6. Scatter the photon. $\theta$ is determined from the $\theta$ lookup table, and $\varphi$ is determined by the rejection sampling method [19]. The electric field is updated by Eq. (1).

7. Perform survival roulette if the weight of the photon is lower than the threshold value. If the photon is dead, go to Step 1, otherwise, amplify the weight of the photon packet appropriately and go to Step 2.

#### 2.3.2 Random number generation

Random numbers are generated by the same method as the pseudo-random number of GPUMCML [20]. In order to generate random numbers in the GPU device, random seeds are loaded from a seed file; the offset is determined by a random number generated from the system time of the computer. Then the random numbers are generated by multiply-with-carry (MWC) random number generators [21]. As with most pseudo-random number generators, the numbers generated by the MWC are periodic and there are certain weak seeds, which should be avoided. However, by choosing appropriate seeds (stored in the seed file beforehand), the number streams generated in the GPU are uncorrelated, mutually independent and uniform enough for Monte Carlo simulations.

#### 2.3.3 Data type and precision

Both single and double precision versions are presented, and as only one macro definition needs to be modified users can easily switch the precision. Because of limitations of GPU hardware, calculations of double precision and mathematical functions are time-consuming [22]. As a result, the double precision version runs much more slowly than the single precision version, as given in Table 1.

While accumulating the electric field values and Stokes vectors, the floating-point numbers are converted to 64-bit integers in the same way as GPUMCML [20]. Although the Fermi architecture supports atomic operation of single precision floating-point, the accuracy of data is seriously decreased for large values encountered during computations. This can be avoided by using 64-bit integers.

#### 2.3.4 Memory access and other optimizations

In the CUDA architecture, there are two kinds of read-only memory accessible by all threads: the constant and texture memory spaces, which are initialized in the CPU and can't be modified by the threads in the GPU. There are several different read-write memories such as register, shared memory, global memory with access to speed register > shared memory > global memory [22]. It is best to use high-speed memory, however, the amount of register and shared memory is rather limited, and therefore, we try our best to optimize the access speed of memory.

Since biological tissue is a forward scattering medium, the scattering angle $\theta$ is not evenly distributed from 0 to $\pi$ ,but rather has a distribution with a higher probability around zero. Thus, storing S1, S2 values in the texture memory results in a high cache hit rate. While looking up a $\theta$ value in the $\theta$ table, a uniform distribution U [1] is used. Therefore, storing $\theta$ table in the texture memory will reduce the performance due to the additional cache space requirement. Thus, the $\theta$ table is stored in global memory.

The Fermi architecture has relaxed requirements on coalescing of the global memory accesses. Coalescing of the global memory accesses will greatly increase the access speed to achieve a higher GPU memory access bandwidth compared with random access. Generally, as the location of the photon in each thread is random, memory accesses are random, and thus

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

the requirements of memory coalescing are not met, resulting in a bottleneck in the simulation. Shared memory is used to temporarily store the electric fields and Stokes vectors, and then save them to global memory with coalesced access.

Figure 2 shows the above process. In a GPU with compute capability 2.0, there are 32 SPs in each stream multiprocessor. Once the electric fields and Stokes vectors have been calculated, these four electric fields (real and imaginary parts of $E_x$ and $E_y$) and four Stokes vectors are stored in registers of each thread. Then the data is copied to shared memory and can be accessed by all the threads in the same GPU block. After all the threads have been synchronized, eight adjacent SPs save eight values of the electric field and Stoke vector calculated by the first SP to global memory at a time, and then the values of the second SP until the values of all eight SPs have been saved. The electric fields and Stokes vectors of any given location are stored continuously in global memory, except for a 64 byte gap, which is filled to make sure the starting address of each access is aligned to 128 bytes, so that the memory accesses of the adjacent eight SPs can be coalesced. By this method, highly efficient memory access is achieved. Since the detection grid size is not dependent on the size of the shared memory used, it has little effect on the performance of the program. The simulation time with a detection grid of 1000 × 1000 is approximately equal to the time with a grid of 100 × 100, as Section 4 describes, there is only an increase of 1.9% in the simulation time when the memory copying time is excluded.

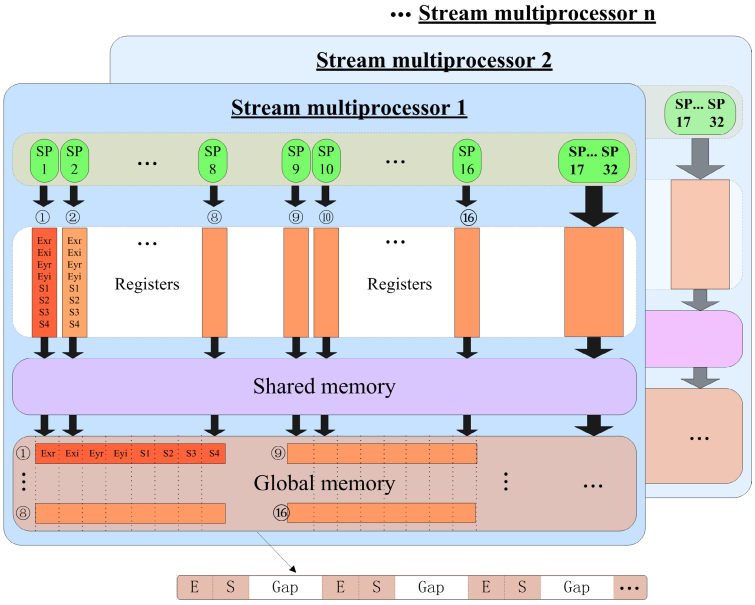


Fig. 2. Schematic diagram of memory access. The eight adjacent SPs access the contiguous global memory. Note that the 64 byte gaps in global memory are filled.

We also perform some other optimizations to accelerate the speed of processing. For example, 8 active blocks per multiprocessor (maximum number for a Fermi GPU) are maintained, which effectively hides access delays. Therefore, the grid size in the kernel should be kept at 8 times the number of stream multiprocessors. Some other optimizations are as follows: optimizing the structure of threads, reducing redundant calculations and memory access, decreasing the size of registers and shared memory used per multiprocessor to maximize utilization, avoiding division and remainder operations, using intrinsic instead of regular functions in the single precision version.

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

2.3.5 Running requirements

As described above, the code is based on CUDA, or rather, Fermi, the second generation CUDA architecture. Therefore, it is necessary to use an NVIDIA GPU with a compute capability above 2.0. A Fermi GPU has also more registers and shared memory per multiprocessor, which allows us to load more active threads at the same time. Because the version of the NVIDIA GPU computing SDK used is 4.2, an updated version of the GPU driver (version R295, R300, or newer) is needed.

## 3. Validation

### *3.1 Speckle pattern produced by an infinitely narrow beam*

In order to validate our program, the instantaneous light intensity $I_x = |E_x|^2$ in the pure backscattering direction obtained by EMC and single-precision CUDAEMC were compared. A model of an infinitely narrow photon beam normally incident on an aqueous solution of polystyrene spheres inside a slab was employed. The same parameters are set in each simulation. A total of 100,000 photons were launched, for a wavelength of incident light of $\lambda_{vac} = 0.515\mu m$ in vacuum. The refractive index of water is $n_{bg} = 1.33$. The mean scattering length of light in aqueous solution is about 2.773 μm with $2\pi n_{bg} l_s / \lambda_{vac} = 45$. The diameter and refractive index of the polystyrene spheres are 0.46 μm and 1.59, respectively. The thickness of the slab is $40 l_s$. A 100 × 100 detection grid with a side length of 20 $l_s$ is used. Note that here the electric fields of the photons at locations beyond this grid, which are recorded as boundary photons, are no longer recorded to get an accurate boundary and increase the parallel computing speed. The absorption effect is ignored, and the maximum number of scattering events is set to 1,000 (the same as EMC).

100,000 simulations (resulting in 100,000 images) are performed to obtain the statistical distribution of the normalized light intensity in the x-axis direction $(I_x / I) \big/ \langle I_x / I \rangle$ at a given location, where $I_x$ is the light intensity in the x-axis direction, $I = \langle I_x \rangle + \langle I_y \rangle = S_1$ is the average intensity in one simulation, $\langle I_x / I \rangle$ is the mean of $I_x / I$ over all the simulations at that location. If the number of photons in the simulation is sufficiently large, the average light intensity $I$ in the location will be a constant and can be neglected. Figure 3 shows that the real part of the normalized complex electric field in the x-axis direction at the location follows a Gaussian distribution; Meanwhile, the light intensity $I_x$ yielded by both EMC and CUDAEMC follow a negative exponential distribution. $I_y$ behaves similarly and is not shown here.

The mean and standard deviation of 10,000 results are calculated, respectively. Figure 4 displays the relative error of mean and standard deviation, showing that the relative error between the results of CUDAEMC and EMC is at the same level as the relative error between two simulation runs of EMC. Note that since the photons beyond the detection grid are not recorded in CUDAEMC, the relative error at the boundary of the detection grid is close to −1.

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

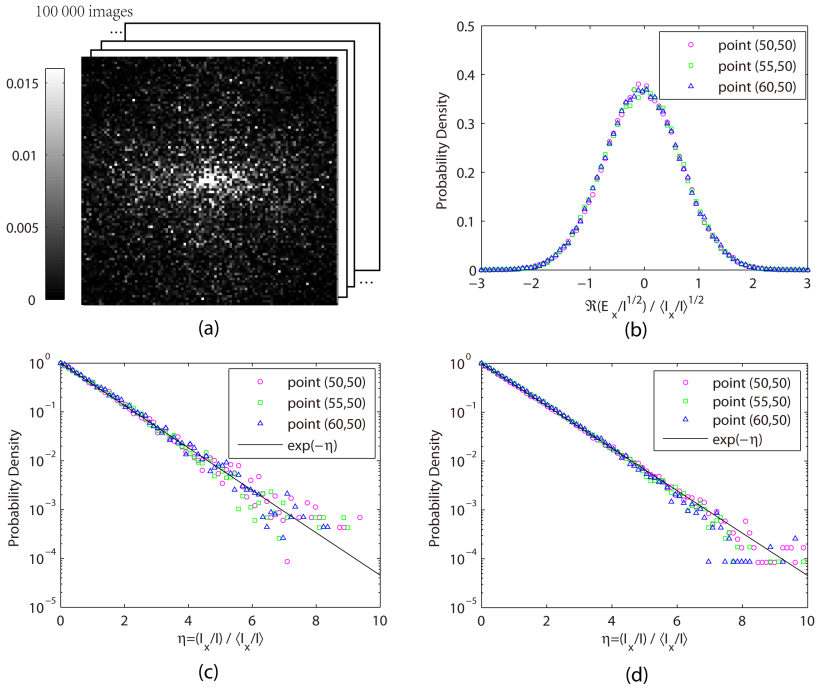


Fig. 3. (a) The exact backscattering light intensity in the x-axis direction spatial distribution is recorded with an infinitely narrow beam incident on the turbid medium. 100,000 simulations are executed to obtain 100,000 images. (b) The probability density function of the real part of the normalized $E_x$ calculated by CUDAEMC, which obeys the Gaussian distribution. (c)(d) The probability density functions of the normalized light intensity in the x-direction at different points of the images calculated from the results of EMC and CUDAEMC, which obey a negative exponent distribution. The parameters of the slab are given in the text

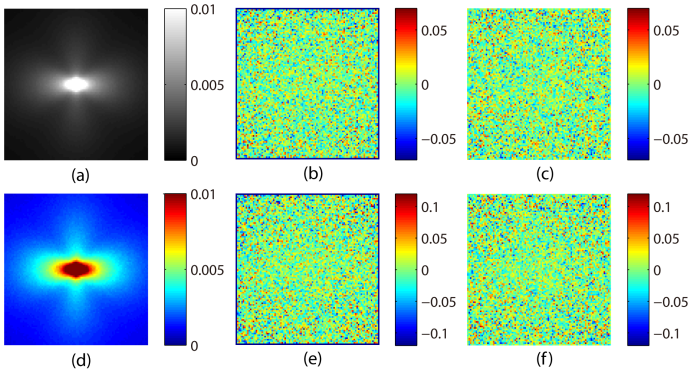


Fig. 4. (a) (d) The means and standard error of light intensity in the x-direction $I_x$ calculated by CUDAEMC. (b)(e) Relative error of mean and standard deviation of light intensity between CUDAEMC and EMC. (c)(f) Relative error of mean and standard deviation of light intensity between two runs of EMC. The side length of each figure is 20 $l_s$ with a resolution of 100 × 100

### *3.2 Speckle pattern produced by a Gaussian beam*

The speckle pattern yielded by a Gaussian beam is displayed in Fig. 5. As we predicted, the normalized light intensity follows a negative exponential distribution. In each simulation, 10

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

billion photons were used, and the waist radius of the beam is kept at 10 $l_s$. All other parameters are the same as in Section 3.1 except using a detection grid of 1000 × 1000.

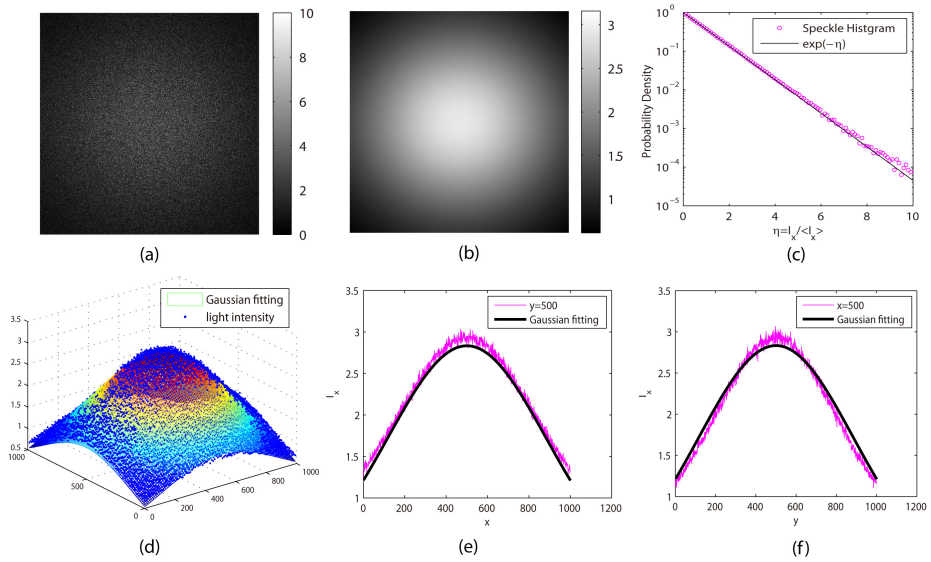


Fig. 5. (a) (b) The instantaneous and average light intensity in x-axis direction. (c) The probability density functions of normalized light intensity in x-direction. (d) The average light intensity approximately follows two-dimensional Gaussian distribution. (e)(f) The cross section of the distribution of light intensity. The side lengths of (a) and (b) are both 20 $l_s$ with a resolution of 1000 × 1000

### *3.3 Mueller matrix*

Figure 6 displays the backscattering Mueller matrix calculated by CUDAEMC and the relative error of the (1,1) element of the Mueller matrix. An infinitely narrow randomly polarized beam composed of 300,000,000 photons is normally incident on an aqueous solution of polystyrene spheres inside a slab. The wavelength of incident light is 0.6328 μm in vacuum. The refractive index of water is taken to be 1.33. The diameter and refractive index of the polystyrene spheres are 2.0 μm and 1.59, respectively. The thickness of the slab is 4 $l_s$. A detection grid with side length 20 $l_s$ is used. The absorption effect is ignored, and the maximum number of scattering events is set to 500 (according to EMC). The relative error between the results of CUDAEMC and EMC is also at the same level as the relative error between two simulation runs of EMC.

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

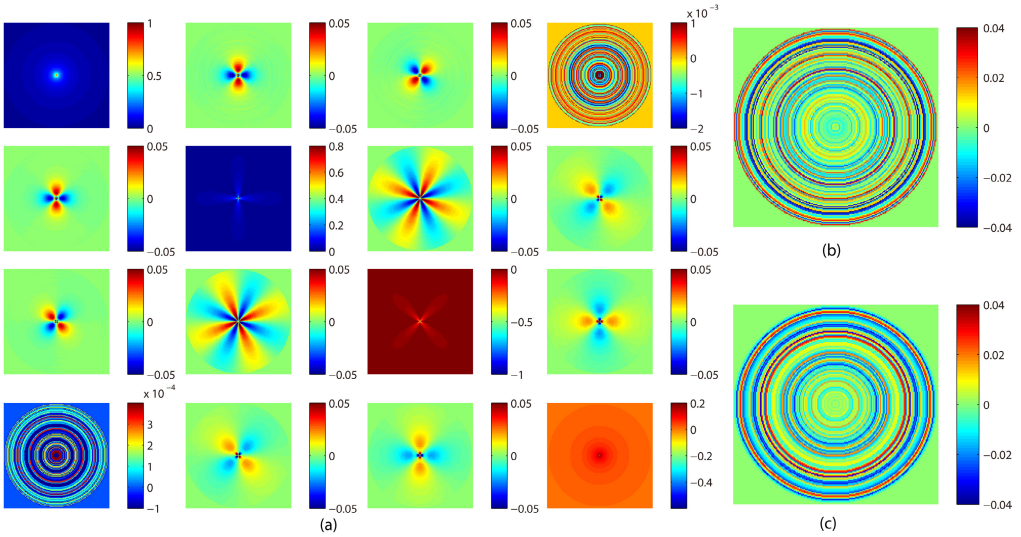


Fig. 6. (a) Mueller matrix obtained by CUDAEMC. (b) The relative error of the Mueller matrix M11 between CUDAEMC and EMC. (c) The relative error of Mueller matrix M11 between two EMC results. The parameters of the slab are given in the text. The side length of each figure is 20 $l_s$ with a resolution of 200 × 200

## 4. Performance

An NVIDIA GTX 480 GPU with compute capability 2.0 was used to evaluate the performance of CUDAEMC. This GPU operates at clock speeds of 1401MHz on the shader cores, 700 MHz on the core. The 1,536MB memory is connected to a 384-bit bus and delivers 177.4 GB/s of memory bandwidth. The GPU consists of 15 streaming multiprocessors, 480 CUDA cores. EMC simulations with the same parameters were run on an Intel i3-2120 CPU (3.3GHz) as a performance baseline. Both programs were optimized in Visual Studio Integrated Development Environment at the –O2 level (maximize speed), and the version of the NVIDIA GPU computing SDK used was 4.2. Note that the EMC program calculates in double precision.

We used two slabs of different thickness to perform respective simulations. Slab 1 and Slab 2 have thicknesses of $4\,l_s$ and $40\,l_s$, respectively. Except for the total number of photons launched, which is 50,000,000, all parameters in the speckle and Mueller matrix simulations were the same as those used in Section 3.1 and Section 3.3, respectively. Note that when using the thinner slab, a lower speedup is achieved. This is because using an extremely small thickness results in a large number of program branches in the GPU, which reduces the parallel computing speed. Here we choose a large value for the number of photons in order to make the test more accurate. The running time and speedup of the simulations are listed in Table 1.

**Table 1. Performance Comparison of Programs**

| | | CPU | GPU(double) | Speedup(double) | GPU(single) | Speedup(single) |
|---|---|---|---|---|---|---|
| Slab 1 ($4\,l_s$) | Speckle | 195s | 3.67s | 53x | 1.09s | 178x |
| | Mueller matrix | 198s | 3.97s | 49x | 1.60s | 123x |
| Slab 2 ($40\,l_s$) | Speckle | 2852s | 42.95s | 66x | 7.69s | 370x |
| | Mueller matrix | 3320s | 50.59s | 63x | 9.14s | 353x |

Figure 7 plots the running time of speckle simulation versus detection grid size for single-precision CUDAEMC. Note that when the size of the detection grid increases, more data

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012

needs to be copied from GPU memory to computer memory, which is the main reason for the increase in running time.

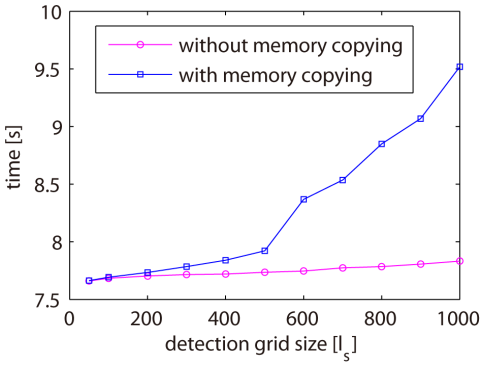


Fig. 7. Simulation time with and without memory copying for different detection grid side lengths.

## 5. Discussion and conclusion

Electric field Monte Carlo simulation provides a tool to simulate the coherence properties of light propagating in a turbid medium, making it possible to investigate the effect of scatterer parameters, and the shape of incident light beam (the original EMC does not provide this functionality) on speckle imaging or some other coherence phenomena. However, it requires intensive computation in order to obtain an accurate result, especially for a finite-size beam simulation. Since the photons move and scatter independently, the computation is easy to be parallelized. The CUDA standard published by NVIDIA Corporation provides a simple tool to realize parallelization on a GPU, which is suitable to accelerate Monte Carlo simulations.

In this paper, we developed an EMC simulation for a finite-size beam of normal incidence, such as a Gaussian beam, circular and rectangular flat beams, and presented a GPU implementation optimized for the Fermi architecture. As noted above, simulating a finite-size beam incident on a turbid medium will obviously increase the amount of computation. When performing simulation on the GPU, several global memory accesses are needed, which cause a bottleneck in the simulation process. Generally, it is very hard to achieve a high computing speed without coalesced access. By rationally designing the structure of the program and data, we effectively increase the probability of memory coalescing and obtain a significant speedup. For a GTX 480 GPU, a maximum speedup of over 370x was achieved when calculating a backscattering speckle pattern to single precision compared with simulation for an Intel i3-2120 CPU. Since the size of shared memory used has little dependence on the size of the detection grid, the program has a relatively stable computing speed. This may be attractive when applying the Electric field Monte Carlo simulation to other fields.

Potential applications of CUDAEMC include investigating the coherence and polarization phenomena of light propagating through turbid media, and investigating the effects which are not included in the traditional numerical speckle simulation, of the parameters of the scattered particles such as albedo and the size parameter on the speckle pattern [23,24]. CUDAEMC may be used for fast speckle simulation of a finite-size beam normally incident on a turbid medium. The CUDAEMC source code will be released on this website: http://bmp.hust.edu.cn/GPU_Cluster/GPU_accelerate_biomedical.html.


## Acknowledgments

This work is supported by the National High Technology Research and Development Program of China (Grant No.2012AA011602), the Science Fund for Creative Research Group

of China (Grant No.61121004), and the National Natural Science Foundation of China (Grant Nos. 30970964, 30800339).

#167113 - $15.00 USD Received 19 Apr 2012; revised 29 Jun 2012; accepted 1 Jul 2012; published 9 Jul 2012