Computer Methods and Programs in Biomedicine 150 (2017) 97–105

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Massively parallel simulator of optical coherence tomography of inhomogeneous turbid media

Siavash Malektaji[a], Ivan T. Lima Jr.[b], Mauricio R. Escobar I.[a], Sherif S. Sherif[a,*]

[a] University of Manitoba, Department of Electrical and Computer Engineering, 75A Chancellor's Circle, Winnipeg, Manitoba R3T 5V6, Canada
[b] North Dakota State University, Department of Electrical and Computer Engineering, 1411 Centennial Boulevard, Fargo, ND 58108-6050, USA

ARTICLE INFO

Article history:
Received 31 March 2016
Revised 31 July 2017
Accepted 7 August 2017

Keywords:
Optical coherence tomography
Light propagation in tissue
Parallel processing

ABSTRACT

Background and objective: An accurate and practical simulator for Optical Coherence Tomography (OCT) could be an important tool to study the underlying physical phenomena in OCT such as multiple light scattering. Recently, many researchers have investigated simulation of OCT of turbid media, e.g., tissue, using Monte Carlo methods. The main drawback of these earlier simulators is the long computational time required to produce accurate results. We developed a massively parallel simulator of OCT of inhomogeneous turbid media that obtains both Class I diffusive reflectivity, due to ballistic and quasi-ballistic scattered photons, and Class II diffusive reflectivity due to multiply scattered photons.

Methods: This Monte Carlo-based simulator is implemented on graphic processing units (GPUs), using the Compute Unified Device Architecture (CUDA) platform and programming model, to exploit the parallel nature of propagation of photons in tissue. It models an arbitrary shaped sample medium as a tetrahedron-based mesh and uses an advanced *importance sampling* scheme.

Results: This new simulator speeds up simulations of OCT of inhomogeneous turbid media by about two orders of magnitude. To demonstrate this result, we have compared the computation times of our new parallel simulator and its serial counterpart using two samples of inhomogeneous turbid media. We have shown that our parallel implementation reduced simulation time of OCT of the first sample medium from 407 min to 92 min by using a single GPU card, to 12 min by using 8 GPU cards and to 7 min by using 16 GPU cards. For the second sample medium, the OCT simulation time was reduced from 209 h to 35.6 h by using a single GPU card, and to 4.65 h by using 8 GPU cards, and to only 2 h by using 16 GPU cards. Therefore our new parallel simulator is considerably more practical to use than its central processing unit (CPU)-based counterpart.

Conclusions: Our new parallel OCT simulator could be a practical tool to study the different physical phenomena underlying OCT, or to design OCT systems with improved performance.

1. Introduction

An increasing number of biomedical applications greatly benefit from optical coherence tomography (OCT) imaging [1–8]. This high-resolution, non-invasive technique is ideal for, but not limited to, imaging soft tissues. Further studies of the underlying physical phenomena and design of novel OCT systems could be carried analytically or through experiments. However, a practical computer simulator of such systems could enable and/or facilitate such efforts. In this paper, we describe a massively parallel simulator of OCT, (which we call OCT-MPS) of inhomogeneous turbid media using a graphics processing unit (GPU) that is more than one order of magnitude faster than its central processing unit (CPU)-based counterpart [9].

Due to its accuracy and applicability to arbitrary media, Wilson and Adam introduced the Monte Carlo (MC) method to solve the Radiative Transport Equation (RTE) that has become a widely accepted approach to model photon migration in tissue [10]. Later Wang et al. [11], developed a well-known MC simulation of light transport in multilayered turbid media (MCML). A broad review of methods to simulate light transport in turbid media can be found in Zhu and Liu [12].

The first MC simulator of OCT imaging was introduced by Smithies et al. in 1998, but it was limited to OCT signals from single layered media [13]. In 1999, Yao and Wang developed an MCML-based simulator of OCT of multilayered media that used an importance sampling method to reduce the variance of simulated OCT signals from a narrow tissue slice [14]. Lima et al.

* Corresponding author.
E-mail address: Sherif.Sherif@umanitoba.ca (S.S. Sherif).

http://dx.doi.org/10.1016/j.cmpb.2017.08.001
0169-2607/

improved this simulator by introducing an advanced *importance sampling* scheme that decreased the computation time of OCT signals from multilayered tissue by two orders of magnitude [15,16]. In 2007, Kirillin et al. developed a simulator of OCT of non-planar multilayered media [17]. In this approach, the boundaries of layers inside the media were modeled as mathematical functions, e.g., sinusoidal functions. This simulator has been used to simulate OCT imaging of human enamel in which its boundaries were modeled as non-parallel planes [7]. Dolganova et al. used this simulator by Kirillin et al. to simulate OCT images of skin with dysplastic nevus [18]. Moreover, Shlivko et al. used this simulator to analyze the structure and optical parameters of layers of skin with thick and thin epidermis [19]. Kirillin et al. also used this simulator to analyze the Speckle statistics in OCT signals. Using the Monte Carlo simulation and experimental results, they showed that Speckle could be modeled with Gamma distribution [20]. Periyasamy and Pramanik developed a Monte Carlo simulator for OCT of multilayered media with embedded objects (such as a sphere, cylinder, ellipsoid, and cuboid) [21] and used importance sampling to reduce the computation time of simulations [22]. In his Ph.D. thesis, Sinan Zhao has developed a Monte Carlo based TD-OCT and FD-OCT simulator [23]. To model media with complex geometry, he subdivided media into cuboidal voxels. He used importance sampling and parallelization to reduce the computation time of his simulator. The disadvantage of using cubical voxel is its inaccuracy in the estimation of the specular (i.e., Fresnel) reflection from tilted surfaces [23].

Recently an MC-based simulator of OCT of inhomogeneous turbid media with arbitrary spatial distributions was introduced by Malektaji et al. [9]. In this OCT simulator, an arbitrary object was represented as a tetrahedron mesh that could model an arbitrary shape with any desired accuracy. An advantage of using tetrahedrons as the building blocks of an arbitrary object is that it minimizes the computation cost of OCT simulation compared to other possible building blocks [9]. This tetrahedron mesh could be obtained using mesh generator applications such as NETGEN [24]. Even though Malektaji et al. used an advanced *importance sampling* scheme to reduce computations, simulation of a *B-scan* of a sphere inside a slab, using 10^7 photon packets, required approximately 360 h on a typical central processing unit (CPU) -based desktop computer [9].

In this paper, we describe a massively parallel implementation of the CPU-based simulator described in [9]; this parallel implementation resulted in a reduction of simulation time by more than one order of magnitude. Therefore, it would be considerably more practical to use our new parallel implementation, i.e., OCT-MPS, compared to its CPU-based counterpart. Our massively parallel implementation runs on graphics processing units (GPUs), using the Compute Unified Device Architecture (CUDA) platform and programming model by NVIDIA [25]. In the following sections, we give an overview of the core concepts of parallel programming with CUDA before explaining in detail the parallel implementation of our simulator. Also, we describe the simulation of OCT imaging of inhomogeneous objects, our GPU memory utilization and our generation of parallel random numbers. We also provide portability guidelines for running our simulator on different CUDA GPUs. Finally, we present simulation results of OCT of nonplanar inhomogeneous media with arbitrary shapes and discuss the reduction in computation time.

2. Simulation of OCT signals from of an inhomogeneous object

Following the procedure to simulate OCT signals from an object consisting of arbitrary shaped regions, in [9], each region is defined with its optical parameters; scattering coefficient μ_s, absorption coefficient μ_a, refractive index n, and an anisotropy factor g. Simulation of light propagation is modeled with a number of photon packets undergoing a random walk inside this object. During such random walks, the photon packets experience absorption and scattering events, where one photon packet splits into two packets traveling in different directions. To simulate OCT signals, a large number of photon packets are launched with the same initial position representing a thin beam incident perpendicular to the surface of the medium. The photon packets are traced inside the medium according to the rules described in references [11,14]. The flowcharts of the process of tracing photon packets inside the medium are shown in Fig. 1.

A collecting fiber, with a specific acceptance angle, $\theta_{\max}$, and radius, $d_{\max}$, is located at the top of the medium to detect backscattered photons. The photons are collected by the fiber probe if their angle and position are within the acceptance angle of the probe. Three types of photons are collected from a depth z by the fiber: (1) ballistic photons that are single scattered, (2) quasi-ballistic photons that are multiply scattered within the coherence length of the optical source, and (3) multiply scattered photons beyond the coherence length of the optical source. Ballistic and quasi-ballistic photons contribute to Class I diffusive reflectivity. The multiply scattered photons beyond the coherence length contribute to Class II diffusive reflectivity. It has been shown that Class II diffusive reflectivity is the main limiting factor in imaging depth of OCT [26,27].

3. Implementation of our OCT massively parallel simulator

The implementation of our OCT massively parallel simulator (OCT-MPS) addresses the high computational load of processing a typically large number of samples required for a suitable accuracy. In our simulator, a large number of samples, i.e., photon packets, are launched simultaneously and traced independently. Thus, we should expect a considerable reduction in computation time due to parallel implementation. In this Section, we describe the CUDA programming environment, the design and implementation of OCT-MPS, including photon packet tracing, random number generation, and memory utilization. We also discuss its portability across different GPUs.

3.1. CUDA programming environment

Compute Unified Device Architecture (CUDA) is a platform and programming model developed by NVIDIA for graphics processing units (GPUs). Its Application Programming Interface (API) offers extensions for many industry standard programming languages, like the C language, with CUDA's accelerated libraries and compiler directives [28]. The CUDA environment allows one to simultaneously develop code intended for both central processing unit (CPU) and GPU.

The basic unit of execution in a CUDA program is a *thread*, which runs independently and concurrently with a big number of other similar threads. CUDA uses an execution model known as Single Instruction, Multiple Thread (SIMT), which allows independence between threads. SIMT allows each thread to have different execution path, separate registers, and possibly execution divergence that would result in different values in instruction address counters [29]. The most common execution divergence occurs with control flow statements (e.g., *if-then-else, switch*), where threads that are not meeting a logical condition are stopped until all other threads fulfilling this condition finish executing this condition [30,31]. To minimize performance penalties, CUDA caches data from stopped threads for fast access. However, avoiding execution divergence and minimizing execution times of all logical conditions are preferable but not always simple to implement [32,33].