Ultra-Fast Hybrid CPU-GPU Monte Carlo Simulation for Scatter Correction in 3D PETs

Kyung Sang Kim and Jong Chul Ye*

***Abstract*—A scatter correction algorithm is very important in 3D PET reconstruction due to large scatter contribution in measurements. Currently, one of the most popular methods is single scatter simulation (SSS), which enables a fast calculation of scattering with a relatively good accuracy; however, the accuracy of SSS is dependent on the accuracy of tail fitting to find a correct scaling factor. To overcome this drawback and to improve accuracy of scatter estimation by incorporating multiple scattering contribution, this paper implements a fast Monte Carlo (MC) simulation that considers accurate photon migration and interactions due to photoelectric absorption and Compton scattering. MC simulator provides a prompt and a scatter data separately; hence, a scaling factor can be calculated by comparing a simulated prompt data with a measured data rather than using a tail fitting, which enables a more robust calculation of the scaling factor. Furthermore, we exploit a hybrid CPU-GPU algorithm using the open multi-processing (OpenMP) and CUDA, which results in 124.3 times faster than using a single CPU.**

I. Introduction

Modern positron emission tomography (PET) systems are designed as fully 3D systems to achieve high sensitivity and enhanced resolution. However, the number of scatter photons significantly increases due to the lack of inter-slice septa. Since scatter affects resolution as well as quantitative aspects of imaging, an accurate scatter correction is essential for a 3D PET.

Clinically, one of the most widely used scatter correction algorithms is the single scatter simulation (SSS) [1] that considers only single Compton scattering effects calculated by the Klein-Nishina formula [2] from many randomly distributed scatter points. Scatter estimation of the SSS algorithm is moderately accurate because about 60-70% of scatters in prompt data are due to single scatter events [1]. To adjust magnitude of scatter distribution, SSS needs to localize a tail region outside of the object, and a scaling factor is obtained by a tail fitting [3]. This procedure is, however, often not robust with respect to a selection of the field of view (FoV), and prone to error under limited number of emission counts.

To address these problems, we propose a fast Monte Carlo (MC) simulation using a hybrid CPU-GPU technique. MC simulations have proven to be the gold-standard method for simulating photon migrations [4]. However, due to the extremely large execution time, they have not been used for clinical applications. Hence, one of the goals of this paper is to develop a fast MC simulation for 3D PETs. In particular, we exploit a hybrid CPU-GPU structure to reduce the idle time of CPU, and to achieve a much higher acceleration factor than by GPU alone. More specifically, our MC is implemented using a novel hybrid CPU-GPU technique based on the open multi-processing (OpenMP) and the compute unified device architecture (CUDA). In order to use both CPU and GPU, function calls need to be implemented asynchronously, where controls are returned to the CPU kernel before completing GPU tasks. In our implementation, two different tasks are assigned to CPU and GPU. The conversion into a sinogram is handled by CPU, whereas photon migrations are performed by GPU. We further accelerate the algorithm using a fast interpolation provided by texture memory in GPU. We also incorporate the MC simulation into our fast OSEM using GPU acceleration, where a pixel-driven backprojector and a ray-driven projector are used to exploit the massive thread architecture of the GPU [5]. Our experimental results with phantom and actual experiments using HRRT and HR+ scanners demonstrate that the proposed MC simulation is fast and more robust than SSS under various situations.

Kyung Sang Kim and Jong Chul Ye* are with the Department of Bio and Brain Engineering, Korea Advanced Institute of Science and Technology, Deajeon, Korea, kssigari@kaist.ac.kr, jong.ye@kaist.ac.kr.

II. Monte Carlo simulation for scatter correction

Monte Carlo simulation considers photon migrations and interactions such as photoelectric absorption and Compton scattering. More specifically, to simulate photon migrations, a radiotracer and an attenuation images are reconstructed from a scatter-uncorrected sinogram and an attenuation sinogram, respectively. We then extract emission positions and their intensities from the radiotracer image, and save them in an emission coordinate table. At each emission position, a photon pair is launched toward arbitrary opposite directions with 511 keV energy after setting their initial values to the intensity at each position. Photon migration path l is then calculated by $l = -\ln(\xi)/\mu$, where μ denotes the attenuation coefficient and ξ is a random number generated from the uniform distribution in the range [0,1]. Since gamma ray photons are within the energy range between 250 keV and 650 keV, Compton scattering is dominant. Thus, our MC simulation mainly considers the Compton scattering, and the differential cross-section for Compton scattering is governed by the Klein-Nishina formula [2]:

$$\frac{d\sigma_s}{d\Omega} = \frac{1}{2} r_e^2 P_e^2 (P_e + P_e^{-1} - 1 + \cos^2(\theta)) \quad , \qquad (1)$$

where $P_e = \left(1 + (E_\gamma / m_e c^2)(1 - \cos\theta)\right)^{-1}$ is the ratio of the scattered to incident photon energy E_γ, r_e is the classical electron radius and θ is the scattering angle, σ_s denotes the

978-1-4673-0120-6/11/$26.00 ©2011 IEEE

total scatter cross-section. After migrations of two photons, the number of counts I is estimated by multiplying the probability of detection corresponding to an LoR as:

$$I = wI_aI_b, \tag{2}$$

where

$$I_a = \cos\theta_{\mathbf{d}_a} \exp\left(-\sum_{n=0}^{N_s}\int_{\mathbf{p}_n}^{\mathbf{p}_{n+1}} \mu_a d\mathbf{p}\right) \prod_{n=0}^{N_s}\left(\frac{d\sigma_s}{d\Omega}\right)_n, \tag{3}$$

and

$$I_b = \cos\theta_{\mathbf{d}_b} \exp\left(-\sum_{n=0}^{N_s'}\int_{\mathbf{p}_n}^{\mathbf{p}_{n+1}} \mu_a d\mathbf{p}\right) \prod_{n=0}^{N_s'}\left(\frac{d\sigma_s}{d\Omega}\right)_n, \tag{4}$$

where w is the initial intensity at an emission position, μ_a is the absorption coefficient, and $\mathbf{u}$ represents the unit direction vector of a photon and $\cos\theta_{\mathbf{d}}$ represents the cross-section of the detector. Here, detector efficiencies are assumed to be 1, N_s and N_s' denote the total number of Compton scattering events for each photon until it reaches a detector. In addition, $\mathbf{p}_n$ is the n-th scattering position, whereas $\mathbf{p}_{N_s+1}$ and $\mathbf{p}_{N_s'+1}$ are the corresponding detector positions. After the photon pair arrives at two detectors, a line of response (LoR) is calculated by assuming that two photons are detected at the same time.

In our MC simulation, a scatter flag records whether Compton scattering events have occurred or not. If the energy level of two detections are valid, two detector positions are converted into an LoR, and rebinned into prompt and scatter sinograms by checking the corresponding scatter flags. A scaling factor is calculated by using a least square fitting between the simulated prompt and the measured sinogram. Scaled scatter sinogram is then subtracted from the measured sinogram for scatter correction.

III. Hybrid CPU-GPU Implementation

In this section, we describe a hybrid CPU-GPU implementation of MC simulation. In our MC simulation, a GPU is used for the execution of photon migrations, whereas a CPU is used for the conversion to a sinogram. Our goal is to assign two jobs separately to a CPU and a GPU so that they can be executed in parallel. Thus, the full resources of the CPU and the GPU can be used and the bottleneck of memory transfer between a CPU and a GPU can be avoided. We employ parallel techniques of OpenMP and CUDA for CPU and GPU, respectively.

More specifically, in a GPU kernel, a thread i is assigned for each emission position $\mathbf{q}_i$ as shown in Fig. 1. Since all kernels can access constant memory as rapidly as they can, we store geometry parameters such as radius, voxel and detector size in the constant memory, and calculate an emission coordinate table composed of emission positions and their intensities from a radiotracer image. Then, we calculate photon migrations until the the photon pair is detected, and change to a LoR format. Using LoRs and a scatter flag, events are classified as prompt and scatter sinograms in a CPU kernel. While a CPU calculates the sinogram conversion, a new task of GPU is started.

To maximally utilize CPU and GPU, we separated two jobs. For concurrent executions between the CPU and GPU, event

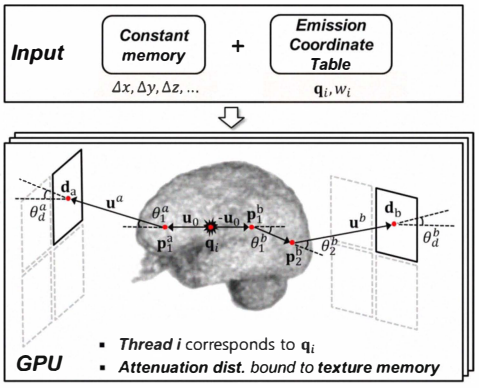

Fig. 1. GPU implementation of a photon migration.

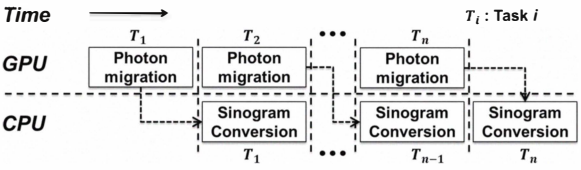

Fig. 2. Hybrid CPU-GPU architecture to perform multiple tasks sequentially.

query and asynchronous memory transfer are considered. GPU can record events at any points in the program and query when these events are completed. Note that when we call a synchronous function, a control waits for a function completion; however, when an asynchronous function is called, a control continues regardless of a function completion. This is why we use asynchronous memory transfer. Specifically, after execution of a GPU task, a control calls an asynchronous memory transfer function and starts the next GPU task regardless of the transfer completion. We then wait for the transfer completion by checking an event query persistently. While a GPU performs photon migrations, memory transfer is completed and the corresponding sinogram conversion is calculated using OpenMP. This architecture maximizes the utility of CPU, which significantly improves the overall performance.

IV. Results

A. HRRT experiments

In an HRRT scanner with an LLD of 400keV, we first evaluated the proposed algorithm using a uniform phantom filled with water. The uniform phantom in figure 3 (a) is a cylindrical phantom filled with water, in which radiotracers are uniformly distributed. To verify the performance, we compared an uniformity ratio. Here, the uniform phantom is divided by inner and outer parts, where the inner part has an area with an $1/8 \sim 2/8$ of the phantom radius and the outer part has an area with $6/8 \sim 7/8$ of the phantom radius. We calculated the averages of the inner and the outer parts; the uniformity ratio is then defined as the ratio between the inner and outer averages. Due to the uniform distribution of radiotracers, the ratio should be 1. However, a scatter uncorrected image reconstruction usually indicates that the average of inner part radius is much

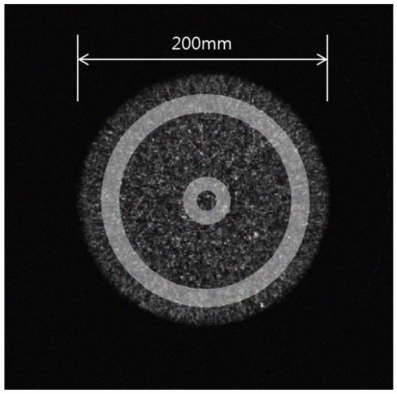

Fig. 3. Reconstruction of an uniform phantom, and the location of inner and outer rings to calculate the uniformity ratio.

Iterations	Inner average	Outer average	Ratio(MC) (inner/outer)	Ratio(SSS) (inner/outer)
1	0.2411	0.2275	1.0590	0.8422
2	0.2219	0.2182	1.0173	1.0418
3	0.2245	0.2217	1.0128	0.9825
4	0.2228	0.2211	1.0076	0.9985

TABLE I
UNIFORMITY RATIOS AT EACH ITERATION USING MC AND SSS.

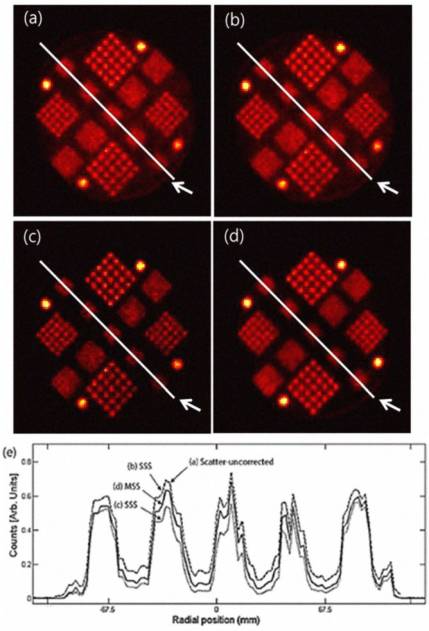

Fig. 4. Reconstruction from (a) a scatter-uncorrected sinogram, (b) a scatter-corrected sinogram by SSS using a tail region of outside of the FoV (out of 580mm), (c) a scatter-corrected sinogram by SSS using a tail region of outside of the object (out of 280mm), (d) a scatter-corrected sinogram by MC, and (e) their cut-views. Diameter of the resolution phantom is 219mm.

higher than that of outer part radius due to the dominant scattering at the center. In Table I, we observed that the uniformity ratio converges to nearly 1 after 2 iterations using MSS and MC, whereas SSS converges after 4 iterations. The result of MSS is also highly comparable to the result of MC.

B. Robustness of scaling parameter

One of advantages of our MC simulation is its robustness in determining a scaling factor. In SSS, a tail region is normally defined as out of FoV before a scatter correction. However, a scaling factor calculation is very sensitive to the selection of a tail region especially when the emission counts are not sufficient. In Fig. 4, we performed experiments with a resolution phantom using an HR+ scanner with LLD of 350keV. Figs. 4(a)-(e) illustrate (a) a scatter-uncorrected image, (b) a scatter-corrected image by SSS using a tail region of outside of the FoV of the scanner, (c) a scatter-corrected image by SSS using a tail region outside of the object, (d) a scatter-corrected image by MC, and (e) their cut-views. Here, the FoV of the HR+ scanner is 580mm and the diameter of the resolution phantom is 219mm. HR+ scanner is a whole body scanner, whereas the resolution phantom is too small to detect scatter counts in the out of the FoV region. Hence, a scaling factor as shown in Fig. 4(b) is not accurate since the emission counts are not sufficient in the tail region. Thus, we found that SSS needs to find an accurate tail region outside of the object to calculate a correct scaling factor as shown in Fig. 4(c). In our MC simulation, tail region is not used; instead, all FoV region of the prompt sinogram is used for a scaling factor calculation. Thus, it is very robust and independent of the choice of tail region. In addition, as shown in Fig. 4(e), SSS using a tail region outside of the object tends to over-estimate the scatter at the center of the image, which results in reduced magnitude at the center in the final OSEM reconstruction image. Since multiple scatter events account for 35% of total scatter for the case of LLD of 350keV (see Discussion), it cannot be simply neglected. Thus, scatter estimation of SSS is usually more concentrated at the center. Our MC simulation is, however, less affected by the artifacts since it takes into consideration of multiple scattering.

C. Execution time for HRRT

The algorithms were implemented on an Nvidia Tesla C1060 system with 4GB of memory with an 800MHz clock and 240 cores with a 1.296GHz clock speed. As shown in table II, we implemented 4 versions of MC simulatior, where OpenMP and GPU are used for CPU and GPU implementations, respectively. To use the hybrid CPU-GPU architecture, asynchronous memory transfer is used. Execution time of the quad cores CPU using OpenMP is 4 times faster, whereas a combination of single core CPU and GPU is 84.4 times faster. Finally, our combination of OpenMP and GPU takes only 9.7 sec, which is 124.3 times faster than single core CPU. In addition, using the 4 times sinogram downsampling, less than 100 millions of photon pairs are required to obtain the scatter distribution.

V. CONCLUSIONS

In this paper, we derived a fast Monte Carlo simulation using a hybrid CPU-GPU implementation with OpenMP and CUDA. Hybrid CPU-GPU implementation reduced the idle time of CPU, and achieved a much higher acceleration than using GPU alone. To maximally utilize CPU and GPU, photon migrations were performed by GPU, whereas sinogram

	MC (sec)	Acceleration
CPU(single)	1205.7	1.0x
CPU(multi)	301.4	4.0x
CPU(single) and GPU	14.3	84.4x
CPU(multi) and GPU	9.7	124.3x

TABLE II
COMPUTATIONAL TIME OF MC USING 100 MILLIONS OF PHOTON PAIRS. PARALLEL TECHNIQUES USING QUAD-CORES CPU AND GPU ARE EMPLOYED. HERE, 'SINGLE' DENOTES THE CALCULATION USING A SINGLE CORE, AND 'MULTI' IS THE OPENMP CALCULATION USING QUAD CORES.

conversion was handled by CPU. Using actual experimental results using HRRT and HR+ scanners, we demonstrated that our MC simulation allows fast and accurate scatter estimation thanks to the robust scaling factor calculation using prompt data rather than scatter data at the tail region, and by considering multiple scattering.

ACKNOWLEDGMENT

This research was supported by the Converging Research Center Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010K001093).

REFERENCES

[1] C. Watson, D. Newport, and M. Casey, "A single scatter simulation technique for scatter correction in 3D PET," *Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, vol. 4, pp. 255–68, 1996.

[2] O. Klein and T. Nishina, "Über die Streuung von Strahlung durch freie Elektronen nach der neuen relativistischen Quantendynamik von Dirac," *Zeitschrift für Physik A Hadrons and Nuclei*, vol. 52, no. 11, pp. 853–868, 1929.

[3] R. Accorsi, L. Adam, M. Werner, and J. Karp, "Optimization of a fully 3D single scatter simulation algorithm for 3D PET," *Physics in Medicine and Biology*, vol. 49, pp. 2577–2598, 2004.

[4] L. Adam, J. Karp, and G. Brix, "Monte Carlo simulation of the scatter contribution in a 3D whole-body PET," *IEEE, Nuclear Science Symposium and Medical Imaging Conference*, vol. 3, pp. 1969–1975, 1998.

[5] K. Kim and J. Ye, "Fully 3D iterative scatter-corrected OSEM for HRRT PET using GPU," *Physics in Medicine and Biology*, vol. 56, pp. 4991–5009, 2011.