GPU-Accelerated First-Order Scattering Simulation for X-Ray CT Image Reconstruction

Sungsoo Ha, Jaewoo Pi, and Klaus Mueller, *Senior Member, IEEE*

***Abstract*— In recent years the GPU has become an increasingly popular tool in various fields. In this paper, we will introduce our preliminary work on first-order scatter simulation in X-ray imaging accelerated by GPU. As this is preliminary work, we explore the GPU accelerated scattering simulation in 2D space and test it with physics-based simulated data. The results are promising.**

I. Introduction

There has been much research on investigating the fast simulation of scattering effects in various fields. It is now well established that restricting efforts to single scatter and simulating it based on ray tracing techniques make it possible for the simulation to be done in a short time [1][2]. Several research groups have proposed hybrid approaches by adding stochastic properties into the deterministic approaches to achieve physical accuracy [3][4]. Furthermore, some have also introduced parallel implementations of scattering simulations to overcome the tremendous amount of computations in the simulation [5][6]. We have pursued the latter where we seek to overcome the challenge by GPU-acceleration [7][8], using their massively parallel computations to meet this challenge.

In general, mapping a CPU-based algorithm to the GPU and achieving 1-2 orders of speedup is not straightforward. It requires a deep understanding of the GPU architecture and its programming model. Furthermore, the CPU-based algorithm often needs to be reordered or decomposed to fit into the GPU. In our case, the ray-tracing based deterministic scattering algorithm [4] is divided into two blocks of separate stages and executed as follows. We first compute the attenuation from the x-ray source to object layers and then use the result in the second stage to perform the scattering simulation. This avoids a lot of redundant attenuation computations that occur when similar rays scatter at different locations in the object.

Sungsoo Ha, Jaewoo Pi, and Klaus Mueller are with the Center for Visual Computing, Computer Science Department, Stony Brook University, NY 11794 USA (e-mail: {sunha, jpi, mueller}@cs.sunysb.edu).

The outline of this paper is as follows. In section 2, we present a brief background on x-ray interaction as well as GPU. In section 3, we provide a detailed explanation of our methods for the first-order scattering simulation on GPUs. The test results are presented in section 4 and section 5 concludes our paper and points to future work.

II. Background

A. X-ray Interaction with Matter

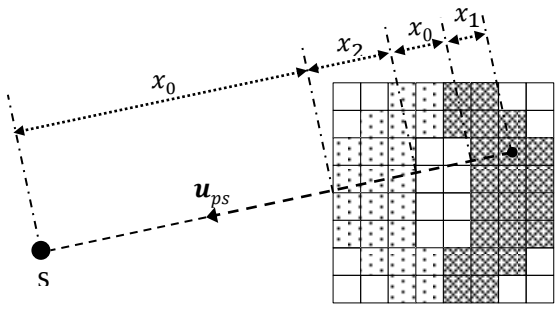
The number of x-ray photons, $N(E)$, is attenuated as it passes through matter in an exponential fashion:

$$N(E) = N_o(E) \exp\left[\sum_i -\mu_i(E) x_i\right] \quad (1)$$

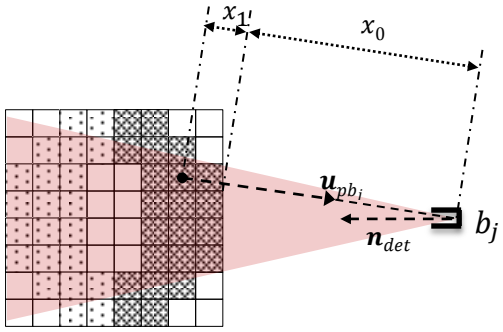
Here, $N_o(E)$ refers to the number of photons emitted from the x-ray source, $\mu_i(E)$ is the linear attenuation coefficient associated with the material i at energy E, and x_i is the total path length through the material i. The linear attenuation coefficient can be re-written as the sum of three individual interaction mechanisms: $\mu(E) = \tau(E) + \sigma_R(E) + \sigma_C(E)$ which are the probabilities of *photoelectric absorption* (τ), *Rayleigh scattering* (σ_R), and *Compton scattering* (σ_C). In photoelectric absorption, the incident x-ray photon transfers its energy to an inner shell electron in the absorbing atom that has a binding energy similar to but less than the energy of the incident photon. In Rayleigh scattering, an incident x-ray photon interacts with an electron and is scattered with a small angle but without loss in energy. Finally, in *Compton scattering* the scattered photon travels in any direction, but with loss of energy. More information on the three interactions can be found in [9]. The form-factor (FF) approximation and the incoherent scattering function (ISF) approximation is the model of Rayleigh- and Compton-scattering, respectively, and most widely used in photon transport simulation [4].

B. NVIDIA GPU and CUDA

We have accelerated the first-order scattering simulation on a NVIDIA 480 GTX GPU with 1.5GB off-chip memory. This GPU has 480 CUDA cores



< 1st stage of scattering simulation >



< 2nd stage of scattering simulation >

s: point source

x_i: ray path length in material i

b_j: j-th bin of 1D virtual detector

u_{ps}: unit vector from a pixel to point source

u_{pb_j}: unit vector from a pixel to j-th bin

n_{det}: normal vector of 1D virtual detector

: material i

: a region of interest for a j-th bin

Fig. 1. The first-order scattering simulation method consists of two stages. The first stage computes the attenuation from source to pixels (top). The second stage scatters photons into bins in a 1D virtual detector (bottom). For convenience x_i only reflects the path length in different materials not the actual length.

organized into 15 streaming multiprocessors (SM) of 32 processors each. Its theoretical computing power is about 1.3 TFLOPS. This GPU, like all modern GPUs, has off-chip memory and on-chip caching mechanisms. Off-chip memory includes global, texture and constant memory and incurs hundreds of cycles of memory latency. It is often the bottleneck of a GPU application. However, texture and constant memory can be cached, replacing the hundreds of cycles of latency with only a few cycles for on-chip cache access. The GTX 480 has a peak memory bandwidth of 177.4 GB/s for its 1.5 GB DDR5 device memory.

The CUDA (Computer Unified Device Architecture) is a C-like API used to program the NVIDIA GPUs. Execution of a task by a CUDA kernel is organized into thread blocks. Thread blocks are organized into a grid. The GTX 480 can have a maximum of 1,024 threads per thread block. Once a thread block is assigned to a streaming multi-processor, it is further divided into 32-thread units called warps and each warp is following same instruction (SIMT: Single Instruction Multiple Thread).

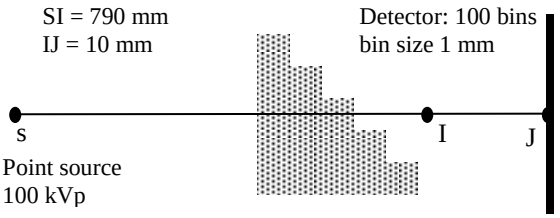
III. Methodology

Our current implementation serializes the computations for the different energy windows, but it parallelizes the computations for the detector bins by assigning each bin a different set of CUDA blocks. The following description refers to one bin and one energy window. To simulate the first-order scattering in a 2D macroscopic object, the object is divided into small pixels. The simulation is comprised of two major stages: (1) the pre-attenuation from the x-ray source to each pixel is computed and stored in memory, and (2) the post-attenuation from each pixel to each bin is computed and combined with the pre-attenuation to obtain the number of Rayleigh- and Compton-scattered photons for the detector bin (Fig. 1). We now describe how the scattering simulation is parallelized.

A. *Early termination in 1st and 2nd stages*

All x-rays from source to pixels (or pixels to a bin) are parallelized. To reduce the amount of computation in the ray traveling, we terminate the sampling of a ray when the current sampling point is outside of the object. The attenuation from the rest of the x-ray is computed based on pre-defined background material and the remained length.

From the view of CUDA programming, the CUDA kernel has 256 (16x16) threads per a thread block. A grid contains $O((width\ of\ the\ object/16) \times (height\ of\ the\ object/16))$ of thread blocks. The configuration is found by considering the tradeoff between processing speed and code scalability. Each pixel in the object has a pre-defined unique object ID such as air (0) and iron (5), and is stored in a 2D texture



Sample object: iron wedge
100 x 100 mm
pixel size 1 x 1 mm

Background: dry air

Fig. 2. Simulation of first-order scattering. A 1D virtual detector is placed behind the sample object. The schematic view of the test setup is not to scale.

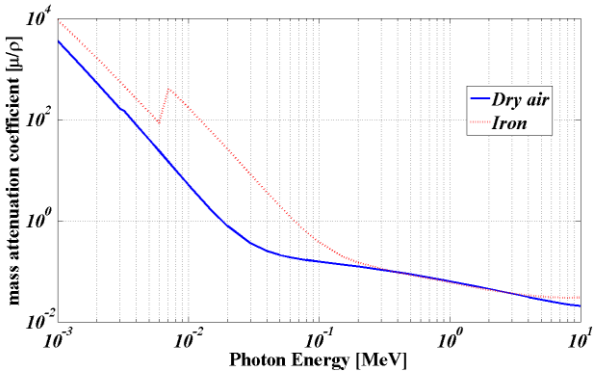


Fig. 4. The mass attenuation coefficient for dry air and iron

which is specifically designed to allow fast 2D memory access and hardware accelerated linear interpolation. The texture memory is also used to store material attenuation coefficients, FFs, and ISFs. This helps to get the data values at a specific energy efficiently. The attenuation coefficient data are obtained from [10] and the FFs and ISFs are from EPDL97 [11]. Before binding these data to texture memory, they are uniformly sampled over a 1keV~1MeV energy range by means of logarithmic interpolation. The constant memory is used to store (effective) atom number and density.

B. *Sum reduction in 2nd stage*

After each thread has computed the number of scattered photons from a pixel to the bin, it stores this number in shared memory. This allows communication of the results among all threads in a thread block. Next we need to combine these pixel photons to obtain the total number of photons arriving in the bin. We use a two-staged approach. First, we use parallelizable sum reduction for computing the total number of scattered photons for each thread block. Then we transfer each block result to the CPU and combine them into the total number of bin photons. This turned out to be faster than a GPU-based block-summation.

C. *Region of interest for a bin at 2nd stage*

To save overall computation at the second stage, we restrict a region of interest (ROI) of the object for a bin

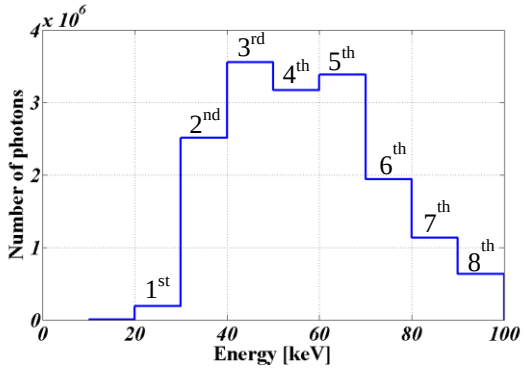


Fig. 3. Discrete x-ray spectrum generated by SpekCalc (100 kVp with 10 keV bin size)

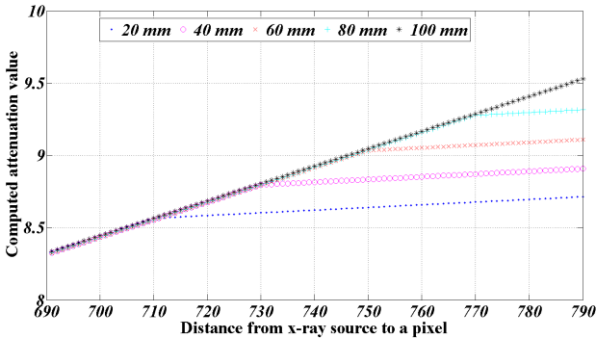


Fig. 5. Profile of pre-attenuation values from 1st stage of the first-order scattering simulation with the test setting at 60~70 keV intervals of x-ray source. Each curve is the profile at the center of a stair of the iron wedge.

(Fig. 1). This can be regarded as a virtual collimator in the detector. Assuming that bins are uniformly distributed over the detector and have the same acceptance angle (between $\boldsymbol{n}_{det}$ and $\boldsymbol{u}_{pb}$, Fig. 1), only a single ROI needs to be computed at the center of the detector. The ROI is repeatedly re-used for all bins with proper translation. The typical time for computing the ROI takes about 1ms, which is considerably less than the time consumed in the two major stages (Table. I). The total number of thread blocks used in the second stage is reduced to $O(size\ of\ the\ ROI/256)$, where the size of the ROI refers to the number of pixels within the ROI and it is typically much less than the total number of pixels of an object.

D. *Parallelizing all bins at 2nd stage*

As the final optimization, the computations for all bins are parallelized so that the first-order scattering simulation with a given x-ray energy window can be done in one GPU kernel invocation. To enable this parallelization, the number of thread blocks is increased. A thread block computes a portion of the number of scattered photons for a bin. This allows to reduce memory transfer from GPU to CPU to the order of $(the\ total\ number\ of\ bins)$.

TABLE I. TIME PERFORMANCE [SEC]

	Value	ROI	Avg. stage 1[4]	Avg. stage 2[4]	Total[5]
Angle[1]	1	0.001	0.003	0.015	0.149
	3	0.001	0.003	0.039	0.349
	5	0.001	0.003	0.064	0.542
Pixel Size[2]	1x1	0.001	0.003	0.082	0.687
	0.5x0.5	0.001	0.007	0.294	2.425
	0.1x0.1	0.015	0.131	5.578	45.766
Bin Size[3]	1	0.001	0.003	0.082	0.687
	0.5	0.001	0.003	0.161	1.322
	0.1	0.001	0.003	0.799	6.429

1. Unit of degree, and fixed 1 x 1mm pixel and 1mm bin
2. Unit of millimeter, and fixed 7 degree angle and 1mm bin
3. Unit of millimeter, and fixed 7 degree angle and 1x1 mm pixel
4. Average time to complete a given task
5. Time to finish scattering simulation with x-ray spectrum (8 windows)

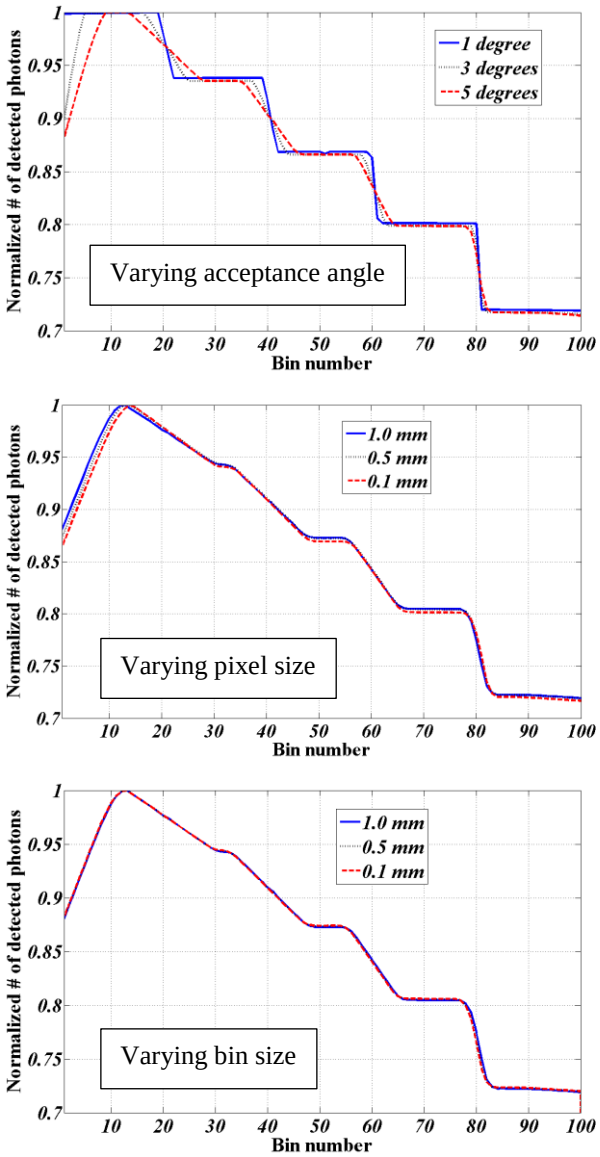


Fig. 6. The number of detected photons with different acceptance angles (top), pixel size (middle) and bins size (bottom). The number is normalized with maximum value at each simulation result for comparison purpose.

IV. Experiment and Results

A. X-ray source

To generate a discrete x-ray source, we used the SpekCalc x-ray spectrum generator program [12]. In this study, we only adjust the peak energy (kVp) and the energy bin size (keV) of the x-ray spectrum. The other parameters of the software are kept as default settings. There are 8 energy windows between 20 and 100 keV (Fig. 3) and the total number of scattered photons is computed as the sum of the photons from all windows. The total time for the simulation is computed as the sum of the time spent in all windows in x-ray spectrum.

B. Test configuration

The x-ray source is located at the origin of the 2D space and the sample object stands 790 mm apart from the source. The detector is placed behind the sample object with 10 mm distance. An iron step wedge is selected as the test sample object. The dimensions of the wedge are 100 x 100 mm. The step width is varying from 20 mm to 100 mm with 20 mm intervals. Dry air is selected as the background material. The detail of this test configuration is illustrated in Fig. 2.

C. Test results

Fig. 5 shows the pre-attenuation values obtained from the 1st stage of the first-order scattering simulation. The value at each pixel indicates the sum of attenuation values along a ray path from the pixel to x-ray source. As a ray travels further and passes through a wider step, the value is increasing. We can also observe that the attenuation value increase more rapidly as the path travels in iron than in air; this is what we can expect in this experiment from Fig. 4.

There are three factors which can affect the time performance and the simulation result: the acceptance angle, pixel size, and bin size. We measure the time performance and the result by varying those parameters. The time performance is shown in Table I. Fig 6 provides comparisons of the total number of detected photons at the detector with different parameter values as shown in Table I. When increasing the acceptance angles, the contrast at the detector decreases which is symptomatic of the scattering effect (Fig. 6 top); while increasing resolution for the detector or the object, does not yield an improved quality of the scattering simulation result (Fig. 6 center and bottom). It is worth noting that in order to compare different numbers of bins, the number of photons in the bins within the coarsest level size is summed.

V. Conclusion And Future Works

In this study we implemented and simulated first-order scattering in 2D space using the GPU. We tested our code with different acceptance angles and different pixel- and bin-sizes. Increasing the acceptance angle to 5 ºgave better scattering effects, while decreasing either pixel or bin size beyond 1.0 was less influential. Using these settings, the measured time performance was about 0.067 seconds for one specific x-ray energy window. For future work, we plan to extend our framework to 3D space and we also plan to verify our results with a well proven scattering simulator and also with directly measured data from real x-ray scanners.

References

[1] D. Lazos, Z. Kolitsi, and N. Pallikarakis, "A software data generator for radiographic imaging investigations," *IEEE Trans. Inf. Technol. Biomed.*, vol. 4, no. 1, pp. 76-79, 2000.

[2] P. Duvauchelle, N. Freud, V. Kaftandjian, and D. Babot, "A computer code to simulate x-ray imaging techniques," *Nucl. Instrum. Methods Phys. Res. B*, no. 170, pp. 245-258, 2000.
[3] F. Inanc and J. N. Gray, "Scattering simulations in radiography," *Appl, Radiat. Isot.*, vol. 48, no. 10-12, pp. 1299-1305, 1997.
[4] N. Freud, P. Duvauchelle, S. A. Pistrui-Maximean, J.-M. Letang, D. Babot, "Deterministic simulation of first-order scattering in virtual x-ray imaging," *Nucl. Instrum. Methods Phys. Res. B*, 222:285-300, 2004.
[5] J. Giersch, A. Weidemann, and G. Anton, "ROSI—an object-oriented and parallel-computing Monte-Carlo simulation for x-ray imaging," *Nucl. Instrum. Methods Phys. Res*. A, 509:151-156, 2003.
[6] F. Inanc, B. Vasiliu, and D. Turner, "Parallel implementation of the integral transport equation-based radiography simulation code," *Nucl. Sci. Eng.*, 137:173-182, 2001.
[7] F. Xu, K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware." *IEEE Trans. On Nuclear Science*, 52(3):654-663, 2005.
[8] F. Xu, K. Mueller, "Real-Time 3D Computed Tomographic Reconstruction Using Commodity Graphics Hardware," *Physics in Medicine and Biology*, 52:3405-3419, 2007.
[9] J. A. Seibert, J. M. Boone, "X-Ray Imaging Physics for Nuclear Medicine Technologists. Part 2: X-Ray Interactions and Image Formation," *J Nucl Med Technol*. 32:139-147, 2004.
[10] J. H. Hubbel, "Photon mass attenuation and energy absorption coefficients from 1 keV to 20 MeV," *Int. J. Appl. Radiat. Isotopes*, 33:1269, 1982.
[11] D. E. Cullen, J. H. Hubbel, L. Kissel, "EPDL97: The Evaluated Data Library, '97 Version," UCRL-ID-50400, 6(5), 1997.
[12] G. Poludniowski, G. Landry, F. DeBlois, P. M. Evans, F. Verhaegen, "SpekCalc: a program to calculate photon spectra from tungsten anode X-ray tubes," *Phys Med Biol* 54(19):N433-N438, 2009.